01-17-01

04 cp #3
0400

169.1919

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	)	
	:	Examiner: N.Y.A.
TIMOTHY JOHN LINDQUIST	)	
	:	Group Art Unit: N.Y.A.
Application No.: 09/730,558	)	
	:	
Filed: December 7, 2000	)	
	:	
For: INVERSE DWT METHOD AND APPARATUS	)	January 24, 2001

Commissioner for Patents
Washington, D.C. 20231

CLAIM TO PRIORITY

Sir:

Applicant hereby claims priority under the International Convention and all rights to which he is entitled under 35 U.S.C. § 119 based upon the following Australian Priority Application:

PQ 4572, filed December 10, 1999.

A certified copy of the priority document is enclosed.

Applicant's undersigned attorney may be reached in our New York office by telephone at (212) 218-2100. All

correspondence should continue to be directed to our address given below.

Respectfully submitted,

Attorney for Applicant

Registration No. 29,296

FITZPATRICK, CELLA, HARPER & SCINTO
30 Rockefeller Plaza
New York, New York 10112-3801
Facsimile: (212) 218-2200

09/730,558

Patent Office
Canberra

I, LISA TREVERROW, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PQ 4572 for a patent by CANON KABUSHIKI KAISHA filed on 10 December 1999.

WITNESS my hand this
Eleventh day of December 2000

Lisa Treverrow

LISA TREVERROW
TEAM LEADER EXAMINATION
SUPPORT AND SALES

CERTIFIED COPY OF
PRIORITY DOCUMENT

S&F Ref: 472541

ORIGINAL

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION FOR THE INVENTION ENTITLED:

Inverse DWT Method and Apparatus

Name and Address of Applicant:

Canon Kabushiki Kaisha, incorporated in Japan, of 30-2, Shimomaruko 3-chome Ohta-ku, Tokyo, 146, Japan

Name of Inventor:

Timothy John Lindquist

This invention is best described in the following statement:

INVERSE DWT METHOD AND APPARATUS

Technical Field of the Invention

The present invention relates generally to the field of inverse transforms, and in particular, to inverse discrete wavelet transforms (IDWTs).

Background Art

Typically, data compression using wavelet techniques is a two step process, comprising a transform phase, during which the wavelet transform of the data set is calculated, and a subsequent coding stage, during which the resultant set from the transform operation is separated into segments, which are then coded using a specific coder. In regard to decompression, the reverse occurs, with coded blocks first being decoded, and subsequently, the inverse wavelet transform being applied in order to generate the final decompressed output.

The inverse discrete wavelet transform (IDWT) is a computationally intensive operation, and current implementations use large memory stores in order to store intermediate results generated during the inverse operations. This intermediate data is generated, and read in an iterative process, as the IDWT process is performed.

Disclosure of the Invention

The present invention is an alternate method of performing the computational process required to process multiple levels of DWT data, in order to produce decompressed output data in scanline, or band order, with little or no intermediate memory requirement.

According to a first aspect of the invention, there is provided A method for performing an Inverse Discrete Wavelet Transform (IDWT) comprising, for a first sub-band level and a subsequent sub-band level in an N level Discrete Wavelet Transform, the steps of:

(i) inverse transforming, using filters having associated filter widths, data from associated sub-bands in the first sub-band level, to form processed data in a corresponding sub-band in the subsequent sub-band level; and

(ii) inverse transforming, using subsequent filters having the same corresponding associated filter widths, the processed data in conjunction with corresponding data from associated sub-bands in the subsequent sub-band level..

According to another aspect of the invention, there is provided an apparatus for for performing an Inverse Discrete Wavelet Transform (IDWT) in accordance with any one of the aforementioned methods.

According to another aspect of the invention, there is provided a computer program product including a computer readable medium having recorded thereon a computer program for performing an Inverse Discrete Wavelet Transform (IDWT) in accordance with any one of the aforementioned methods.

Brief Description of the Drawings

A number of preferred embodiments of the present invention will now be described, with reference to the drawings, in which:

Fig. 1 is a representation of a three level DWT/IDWT process performed on a one-dimensional data set;

Fig. 2 illustrates the DWT process in Fig. 1 at a data-bit level;

Fig. 3 depicts the IDWT process in Fig. 1 at a data-bit level;

Fig. 4 illustrates an up-sampling/convolution process used in the IDWT process in Fig. 1;

Fig. 5 illustrates data requirements at different levels in an IDWT process;

Fig. 6 illustrates the aforementioned data requirements in a preferred embodiment of the present invention;

Fig. 7 depicts "sliding" filters used in relation to Fig. 6;

Fig. 8 illustrates a multi-level IDWT process according to Fig. 6;

Fig. 9 depicts the cascaded nature of the IDWT process in Fig. 6;

Fig. 10 illustrates the use of "virtual" computational blocks in the preferred embodiment;

Fig. 11 illustrates a one-level DWT process for two dimensional data;

Fig. 12 illustrates a prior art embodiment for performing a two-dimensional IDWT in relation to Fig. 11;

Fig. 13 illustrates a preferred embodiment for performing a two-dimensional IDWT according to the present invention;

Fig. 14 is an arrangment of part of an inverse two-dimensional separable DWT transformer in accordance with a first preferred embodiment;

Fig. 15 shows a vertical filter arrangement which can be used in Fig. 14;

Fig. 16 shows a horizontal filter arrangement which can be used in Fig. 14;

Fig. 17 depicts an IDWT transformer arranged to access and output multiple lines of data;

Fig. 18 shows horizontal a filter arrangement suitable for use in Fig. 17;

Fig. 19 depicts use of "virtual" computational blocks in relation to Fig. 13;

Fig. 20 illustrates a multiplexer used to form the virtual blocks in Fig. 19;

Fig. 21 depicts a multi-stage pipeline which can be applied in relation to Fig. 13;

Fig. 22 illustrates a context store for use in the arrangement of Fig. 21; and

Fig. 23 is a schematic block diagram of a general purpose computer upon which the preferred embodiment of the present invention can be practiced.

Detailed Description including Best Mode

Where reference is made in any one or more of the accompanying drawings to steps and/or features, which have the same reference numerals, those steps and/or features have for the purposes of this description the same function(s) or operation(s), unless the contrary intention appears.

In the context of this specification, the word "comprising" means "including principally but not necessarily solely" or "having" or "including" and not "consisting only of". Variations of the word comprising, such as "comprise" and "comprises" have corresponding meanings.

The principles of the preferred method described herein have general applicability to inverse discrete wavelet transforms in any number of dimensions. For ease of explanation however, the steps of the preferred method are described primarily with reference to one-dimensional and two-dimensional data sets. It is not, however, intended that the present invention be limited thereto.

Fig. 1 illustrates a three level, forward and reverse, discrete wavelet transform process, in regard to a one-dimensional data set 100. The data set 100 is designated as being at the "zero" level, this being depicted by the numeral 102. The first level transformed data is similarly depicted by the numeral 108, with the second and third level transformed data being depicted by the numerals 110 and 112 respectively. In the inverse transform direction, the third level data 112 is inverse transformed to form second level data 120, thereafter being inverse transformed to form first level data 122, and finally

being inverse transformed to reconstitute the original data 124 at the zero level. In the forward transform direction, the data set 100 is transformed to form two first level sub-bands 104 and 106, one of which, ie. 106, being further transformed to form two second-level sub-bands 126 and 128. Sub-band 128 is yet further transformed to form sub-bands 114 and 116, this representing a third level of transformation. In the inverse transform direction, the third-level sub-bands 114, 116 are inverse transformed to produce a second-level sub-band 118, which, together with a sub-band 130 presently being held in memory, are inverse transformed to a first-level sub-band 132. In the inverse IDWT process, the H sub-bands are typically stored in a buffer memory, as can be seen at the third data level 112 where H1, H2 and H3 are seen to be available. Since the forward DWT process formed all the H sub-bands, these are, of necessity, held in buffer storage. However, in the IDWT direction, the need to hold a particular L sub-band in intermediate storage until the next stage of the IDWT process takes place, is undesirable. It is the production of the intermediate L sub-bands which requires the intermediate memory during IDWT computation. Thus, for example, H3 and L3 (ie 114 and 116 respectively) produce L2 (ie 118), and L2 must be held (ie stored) in intermediate storage to be processed with H2 (ie 130) in order to produce L1 (ie 132). Finally, the aforementioned sub-band 132 is inverse transformed together with a sub-band 134, presently held in memory, to re-constitute, either exactly or approximately, the original data 124.

Existing approaches process DWTs one level at a time, writing intermediate results to memory. The same comment applies equally to the IDWT process. For example, when performing the inverse transform from the second level 120 to the first level 122, the sub-band 118, held in intermediate memory, is inverse transformed together with the sub-band 130, the latter being held in buffer memory. Similarly, in proceeding from the first sub-band level to the zeroth sub-band level, the sub-band 132, also held in

intermediate memory, is inverse transformed together with the sub-band 134, the latter being held in buffer memory. The same comments in regard to H sub-bands being available, while L sub-bands require intermediate memory applies as was noted previously.

Considering the IDWT process, since data is up-sampled in each successive IDWT level, the amount of intermediate memory required increases with each level. The memory required for sub-bands 114 and 116 at the third level 112 can be intuitively related to the lengths of the line segments used to represent the sub-bands. It is seen that at the next (ie. second) level 120, the line segment representing the sub-band 130 presently held in storage is twice as long as each line segment representing the sub-bands 114, 116, indicating that the memory storage requirements at the sub-band level 120 are twice as high on a per sub-band basis, as at the previous (ie. third) sub-band level 112. At each level, sufficient memory must be available to store the outputs associated with that level, noting that the memory requirements double (for the one-dimensional case) on a per level basis as the process progresses through the levels. In the two-dimensional case, memory requirements quadruple on a per-level basis, and so on. It is further noted, that the process of storing data in intermediate memory, and retrieving it again at the next level, adds delays in the form of write and read access delay, for each data item which is generated at one level, and used in the next level.

Fig. 2 illustrates a single-level embodiment for a forward, ie. DWT process (the "forward" direction being depicted by an arrow 226) in somewhat more detail. A low pass filter 204 and a high pass filter 206 being centred according to the arrows 210 and 208 respectively are applied to a data set 200. Each filter thus convolves data from the data set 200, the output from the low pass and high pass filters being depicted by arrows 214 and 216 respectively. The low pass filter 204 and high pass filter 206 are centred at

points 2n and 2n+1 in the data set 200, and produce data points "n" in the respective sub-bands 218 and 220. It is further noted that the length 202 of the data set 200 is twice as long as the respective lengths 222, 224 of the respective sub-bands 218 and 220. This "halving" of data length derives from the down-sampling inherent in the DWT process. The filters 204 and 206 are "slid" along the data set 200 in 2 data-point increments, the two-point increment being indicative of the downsampling process, in a direction depicted by an arrow 212.

Fig. 3 illustrates a single-level embodiment of an IDWT or inverse transform process, the inverse nature of the transform process being depicted by an arrow 312. In this case, a low pass filter 306 and a high pass filter 304 are centred as shown on bits "n" and "n-1" of the sub-bands 302 and 300 respectively. Convolution of these filters with the data about their respective centres results (see Fig. 4 for more detail), as depicted by arrows 316 and 314 respectively, after addition in an adder 318, in an output data point "2n" in the resultant inverse-transformed data set 310. The filters 306, 304 are slid in single data-point increments in a direction depicted by an arrow 308.

Fig. 4 illustrates the single-level inverse transform operation described in relation to Fig. 3 in somewhat more detail. Considering a low pass filter 414, it is seen that the length of the filter is 7 data points long. Similarly, a high pass filter 416 is seen to have a length of 9 data points. Due to the up-sampling process inherent in the inverse DWT, the 7 bits required by the low pass filter 414 are derived, ie, the filter is "filled" by up-sampling 4 bits (depicted as 400), the up-sampling process inserting a "0" in between points in the set of data 400 in order to produce the 7 bits required for the low pass filter 414. This up-sampling process is depicted by a dashed arrow 420. Similarly, a data set 402 is up-sampled as depicted by a dashed arrow 418 in order to "fill" the high pass filter 416. Each of the aforementioned filters convolves their respective up-sampled data set,

the results of the two convolutions being added in an adder 406, resulting in an output point 408. The convolution computation is expressed mathematically in Equation (1) in relation to Fig. 15.

In Fig. 5, the IDWT data requirements at each level of transformed data which is necessary to produce a single output point is considered. A single output point 524 at level 0 requires enough data from the first level sub-band to satisfy the requirements of the kernel of the IDWT process, ie. to "fill" the relevant filter. Considering the present embodiment using a low pass filter having nine taps, the output point 524 requires that the filter represented as 526 be filled with nine data points. The filter 526 is centred on a data point 522. It is noted that the data points depicted by an "X", eg. 516, are "real" data points, whereas data points represented by a "0" are not real data points, but rather zeros which are inserted by the up-sampling process. Therefore, the filter 526, while needing the total of 9 points in order to produce the output point 524, in reality only requires five real data points as shown by the "X"s. From another perspective, an output point 524 being centred on the filter centre point 522 requires a set of four data points 520 on either side of the filter centre point 522. Therefore, a level one data point 516 being centred on a level 2 data point 518 requires the set of four data points 512 on the right hand side of the filter centre data point 518. Going up a level, the extreme right hand data point 510 at level 2, being centred on a level 3 data point 508 requires that the filter centre point 508 have a set of four data points 506 available. Moving yet one level up, a level 3 extreme right hand real point 528, being centred on a level 4 data point 504 requires that the level 4 data point 504 have four data points 502 available. It is thus seen, by the time we consider level 4, that the width of the overall data set required to support the original output point 524 is no longer increasing in size. In fact, looking at level 5, it is seen that a centre data point 530 requires 7 data points on either side in order to satisfy the

production of the final output point 524 at level 0. Furthermore, the data point 530 requires only three real data points on either side, ie. a total of seven real data points is required. This is also the case for all subsequent levels beyond level 5. The fact that the required data set width becomes bounded at a certain point, or rather a certain level, can be exploited in a multi-level IDWT in order to produce output points in a raster, or band fashion, using a pipe-line process and fixed data-width filters, without significant intermediate memory storage requirements. This is explained in relation to Figs. 6 and 7.

In Fig. 6, since for all levels beyond level one in Fig. 5, seven real points are required, it is expeditious to standardise on this number of points at all levels including level 1. Since zeros at the extremities of the data sets are inserted by the up-sampling process without requirement for real data from the preceeding levels, it is seen, in a static situation where the filters are not "slid" along the data set as described in relation to Fig. 2, that the use of seven real data points at each level allows seven output points 610 to 602 to be produced, these being centred on data point 600. It is found, however, that this configuration can be improved by using eight points at each level, thereby producing eight output points. When using the eight data point configuration, the filters required are constant in size, ie they require a constant number of real points from the preceding level, as they are slid along the data points in question. If, however, the seven point configuration is used, the filters at the various levels typically oscillate in size by approximately one real data point, which complicates the implementation. In conclusion therefore, for any number of levels, extraction of eight data points at each level, when considering the particular filter described and taking into account the up-sampling process, allows eight output points to be produced (see Fig. 7).

If the number of levels exceeds the capacity implemented in hardware, the computation can be folded, ie a multi-stage pipeline can be used. This is described in more detail in regard to Figs. 20 and 21.

Fig. 7 illustrates a three level IDWT process, still in relation to the one-dimensional data set. Considering a first set of eight output points 724, we see that filters 718, 712 and 710 are filled with the appropriate eight points of real data plus corresponding upsampled zeros, at levels one, two and three respectively. The convolution outputs of these three filters in tandem produce the set of output points 724. Thereafter, the level one filter 718 is slid to the right to a position depicted by 720. The level two filter 712 is similarly slid to the right as depicted by 714, and the level three filter 710 slid to the right as depicted by 706. The initial lateral position of the set of filters is depicted by a dashed line 700, whereas their lateral position is depicted by a dashed line 702 after the sliding operation just described. The three filters 706, 714 and 720, produce a second set of eight output points 726 by convolution of the data points in the filters. By sliding the respective filters to alignment positions depicted by a dashed line 704, a third set of eight output points 728 is produced. To summarise the implication of the embodiment shown in Fig. 7, it is noted that the use of fixed-width filters at each and every band of the multi-level IDWT data set results in production of a band of data at level 0. The practical implication is that instead of utilising the prior art level-by-level method of performing the IDWT, a pipeline process can be implemented, whereby output data can be produced in raster, or band fashion, using a pipeline process of fixed data width, without the use of significant intermediate storage.

Fig. 8 presents a block representation, and depicts how an output point 830 is produced by a tandem arrangement of computational blocks 850, 852 and 854. The block 850 comprises two computational sub-blocks 808 and 816, each of these sub-blocks

providing for up-sampling and convolution of respective data points from sub-bands 804 and 806. The up-sampled convolved data points are added in an adder 810, and together with respective data from a sub-band 802, are directed to the next level computational block 852. The output of block 852 is directed as depicted by an arrow 822 along with data from the next sub-band level 800 as depicted by an arrow 856, to the final computational block 854, which in turn produces the output point 830. Each computational sub-block 808 comprises an up-sampling/convolving process. The convolution, as described in, for example, Fig. 7, is performed by an appropriate IDWT synthesis filter. The up-sampling process is easily performed by inserting zeros between real data points as appropriate. Each computational sub-block 808 is slid to the right as depicted by an arrow 848, in a similar manner to that described in Fig. 7. This same sliding operation is performed for all computation blocks 826, 818, 816 and 808 as depicted by arrows 840, 842, 846 and 848 respectively. The outputs of each computational block 850 are fed in parallel cascade, directly, and without substantial intermediate memory, to the computational block 852 at the next level, and so on. Thus, for each pair of levels, for example, level 3 (ie. 832) and level 2 (ie. 834), a set of sub-bands 804, 806 is inverse transformed by the computational block 850 to form data, depicted by an arrow 812, at a second sub-band level 812, which together with associated sub-band data 802 is again subjected to the IDWT process as performed by the computational block 852.

Fig. 9 presents a block representation of the arrangement in Fig. 8, where a computational block 906, receives input sub-band data 902, and 904 from a block buffer 900. The computational block 906 feeds an output 918 to a computational block 922 at the next level, which together with an input 920 from the block buffer 900 produces data 924 for the next sub-band layer, and so on. Finally output data 916 is produced. The

arrangement in Fig. 9 shows a computational block eg. 906, being associated with each level. It is recalled from Fig. 6, that a single data point at a given level is equivalent to two data points at the next lower level because of the up-sampling process in the IDWT. Therefore from a computation perspective, a computational block 914 at level 1 performs twice as many computations per unit time as the equivalent computational block 908 one level up ie. at level 2. Generalising this principle, if the computational block 914 is defined as performing computations 100% of the time, then the computational block at level 2 will be performing computation for only 50% of the time, and the computational block on level three, for only 25% of the time. Therefore, considering a j level transform, the computational blocks for levels 2 through j, are "busy" for a total time which can be represented mathematically by the following series:

$$1/2 + 1/4 + 1/8 + 1/16 + \ldots + (1/2)^{j-1} .$$

The above series converges to "1", and therefore we conclude that one computational block is required for level 1 processing, and a second computational block can be shared by all other levels, from level 2 through to level j.

The aforementioned principle is utilised as shown in Fig. 10, where a computational block 1000 is required at level 1, and a second computational block 1002 is shared among all the remaining levels, the block 1002 being switched between levels, to act as a "virtual" computational block as depicted, for example, by a dashed block 1006. This represents a significant saving in computational hardware, at the cost of switching hardware required to switch the block 1002 between levels, and possibly, a small memory 1004 to save the state information from each computational block 1002 while it is being used at another level, eg. 1006.

Fig. 11 depicts a single level DWT for two-dimensional data. A data set 1100 is first "row-transformed" as depicted by an arrow 1102, the high and low pass filters being translated two data points at a time as described in relation to Fig. 2, and then moved to a next line 1106 as depicted by a retrace arrow 1104 after each row 1102 of data is transformed. Once the data set 1100 is fully transformed, as depicted by a dashed arrow 1108, it is represented, by two data sets 1110 and 1116, having down-sampled row-transformed data 1114 and 1116 respectively. These transformed data sets 1110 and 1112 are then, as depicted by an arrow 1118, subjected to column data transformation as depicted by an arrow 1120, each column being fully transformed, and the transform filters then being moved to another column 1124 as depicted by an arrow 1122. At the end of the column transform process, the data is transformed as represented by a dashed arrow 1130, into a set of four sub-bands 1132.

Considering a transform with j levels, at each level other than level j, the data for an LL sub-band 1138 will be generated by the application of the IDWT at a previous level, the data for the previous sub-bands being retrieved from associated decoded blocks being held in memory, similar to the block buffer 900 in Fig. 9 for the one-dimensional case.

Fig. 12 presents, as a basis for further discussion, a description of an IDWT process at a particular level in a conventional memory based approach for two-dimensional data. Processing takes place in a number of stages. Stage A shows the four sub-bands 1200 at a particular level. All sub-bands 1200 have the following operations performed upon them: an up-sampling by two and a filtering (ie. convolution) in the vertical direction as depicted, for example, by a computational block 1204. Stage A is completed when all the results for each sub-band in 1200 have been written to a memory, eg. 1208, 1210. Stage B retrieves the results associated with the previous sub-bands 1200

from memory 1210, 1208, which are added pair wise in adders, eg. 1212, and the results stored in a memory 1214. Stage C is completed when all the results for each data set have been written to memory 1216 and 1224. Stage D retrieves the results associated with each data set from memory 1216, 1224, these being added in an adder 1218 to form the final data set 1220. At this stage, the data for the LL sub-band for the next level has been generated, and the whole process may be repeated at the next level. The aforementioned process occurs repeatedly at each level, until all levels are processed and the final output data is produced.

It is apparent that operation to and from memory imposes a penalty in relation to the total time required to fully apply the inverse transform. Since each intermediate data point requires a write cycle into memory and a later read cycle to retrieve it, this penalty increases with the number of levels being considered. The memory requirements also increase at each level.

Recalling Fig. 6 however, and extrapolating the one-dimensional concept to the two-dimensional case, an area of 8 × 8 "real" points at a level, will be seen to satisfy the requirements of the next level, thereby generating 8 × 8 "real" points at that next level. This principle holds for all levels. Therefore a 8 × 8 area if provided at each level, will produce 8 × 8 output data at level 0.

Fig. 13 depicts a preferred embodiment for the two-dimensional case which takes advantage of the aforementioned feature. The Figure describes an IDWT computation block at level 1, ie. the level which produces output data at level 0. The computational block takes the same form at each level however. In addition, we recall the fact that resource sharing is possible so that computational blocks can be shared between levels.

The embodiment in Fig. 13 is described in relation to a 7 X 7 point computation, however it is not limited thereto. The embodiment produces a column of seven output pixels 1324 on each occasion that a column of input data eg. 1302 is clocked into parallel convolvers eg. 1304, once sequential convolvers eg. 1316 have filled with data. The combination of a parallel convolver 1304, associated adders 1332, and associated sequential convolvers (eg 1316), which are contained in a dashed box outline 1344, are referred to as a two-dimensional separable IDWT transformer, which is described in more detail in relation to Figs. 14 to 16, as will the interpolator function which performs upconversion which is not shown explicitly in Fig. 13.

To perform the operations in the vertical direction, four parallel convolvers eg. 1304 are used, one per sub-band. Each such convolver 1304 simultaneously produces the results of the vertical kernel (ie. filter) applied separately at seven input "real" points. Thus, instead of the filter "sliding" as depicted in Fig. 7, data is loaded into the parallel convolvers in parallel. The total number of points 1302 required to produce the seven outputs 1340 centred around these points 1302 is 15 points, ie. the seven real points and eight zeros that are the result of up-sampling the data. The zeros need not be read from memory, and may be inserted by suitable implementation of the convolver itself.

Thus from each sub-band, a column of seven points 1302 is input. Corresponding intermediate points for the LL and LH sub-bands, eg. 1306 and 1308 respectively, are added in an adder 1332. The output of the adder 1332 is fed into the sequential convolver 1316 which operates in the horizontal direction, one sequential convolver 1316 being required per output point in the data set 1340 which is output by the parallel converter 1304. As data is stepped into the horizontal convolvers eg. 1316, a line of output data 1318 is progressively generated. Finally corresponding data , eg. 1318 and 1334, from the left and right horizontal serial convolvers 1316 and 1336 are added in an

adder 1338. The result is to produce seven lines 1342 of fully convolved data that are built up sequentially, meaning that all lines are produced in parallel (ie. 1324), concurrently, one point at a time in raster order, at a rate that the serial convolvers, eg. 1316, have data fed into them. The output from the final adder stage 1338, is in fact the LL sub-band data required by the next level.

When the computational block depicted in Fig. 13 is applied at other levels e.g., level 2, it is noted that once a level e.g., level 2, has produced seven columns of LL data 1324 for the next lower level e.g., level 1, further LL data 1324 from level 2 is only generated when required by the next level i.e., level 1. This relates to the aspect of up-sampling between levels, and the consequent differing rates of computational requirements at each level.

The present two-dimensional embodiment as described in relation to Fig. 13 progresses in one-step data point increments from left to right in the horizontal direction, and thereby generates a band of data eg. 1324 in raster order, the band being seven data points wide. At the end of a traversal across the sub-bands in a horizontal direction, the process steps down seven rows in the vertical direction and re-traces in a right to left direction, in order to continue the inverse transformation process.

Fig. 14 shows a schematic representation of part of an inverse two-dimensional separable DWT transformer. The arrangement 1600 comprises a vertical filter 1602 feeding a horizontal filter 1604 via an adder 1606 and an interpolator 1608. The arrangement 1600, as distinct from typical wavelet transformers, has no large-scale intermediate memory.

For ease of explanation, the operation of the arrangement of Fig. 14 is described with reference to a wavelet transform sub-band having a plurality of transform coefficients $x_{r,s}$ arranged in a plurality of rows r and columns s. The vertical filter 1602

has multiple parallel inputs 1601 and one output channel 1605. In this particular example, the vertical filter 1602 has nine taps and nine parallel inputs 1601 but is not intended to be limited thereto. Similarly, the horizontal filter 1604 has nine taps but is not intended to be limited thereto.

At any one clock cycle, the vertical filter 1602 takes as its input nine transform coefficients $\{ x_{i-4,j}, x_{i-3,j}, x_{i-2,j}, x_{i-1,j}, x_{i,j}, x_{i+1,j}, x_{i+2,j}, x_{i+3,j}, x_{i+4,j} \}$ of the wavelet transform sub-band. The vertical filter 1602 then calculates and outputs one intermediate transform coefficient $y_{i,j}$ based on these input transform coefficients. That is, the filter 1602 undertakes a one-dimensional inverse transform in the vertical direction of the coefficients $\{ x_{i-4,j}, x_{i-3,j}, x_{i-2,j}, x_{i-1,j}, x_{i,j}, x_{i+1,j}, x_{i+2,j}, x_{i+3,j}, x_{i+4,j} \}$. Thus, the resultant intermediate coefficients $y_{i,j}$ are one-dimensional transform coefficients in the horizontal direction of the image at position i,j.

At the next clock cycle, the vertical filter 1602 takes as its input the nine transform coefficients $\{ x_{i-4,j+1}, x_{i-3,j+1}, x_{i-2,j+1}, x_{i-1,j+1}, x_{i,j+1}, x_{i+1,j+1}, x_{i+2,j+1}, x_{i+3,j+1}, x_{i+4,j+1} \}$ of the wavelet transform sub-band and outputs another intermediate transform coefficient $y_{i,j+1}$. At the next clock cycle, the vertical filter 1602 takes as its input the nine transform coefficients $\{x_{i-4,j+2}, x_{i-3,j+2}, x_{i-2,j+2}, x_{i-1,j+2}, x_{i,j+2}, x_{i+1,j+2}, x_{i+2,j+2}, x_{i+3,j+2}, x_{i+4,j+2} \}$ of the wavelet transform sub-band and outputs another intermediate transform coefficient $y_{i,j+2}$. The vertical filter 1602 continues in this manner for each clock cycle until the end of the row i. The vertical filter 1602 then takes as its input the nine transform coefficients $\{x_{i-3,0}, x_{i-2,0}, x_{i-1,0}, x_{i,0}, x_{i+1,0}, x_{i+2,0}, x_{i+3,j}, x_{i+4,0}, x_{i+5,0} \}$ and continues in a similar manner as the previous row. As can be seen, the vertical filter 1602 receives as input groups of coefficients in sequence. Each group $\{ x_{i-4,j}, x_{i-3,j}, x_{i-2,j}, x_{i-1,j}, x_{i,j}, x_{i+1,j}, x_{i+2,j}, x_{i+3,j}, x_{i+4,j} \}$ comprises nine adjacent coefficients arranged in the vertical direction of the image, four on each vertical side of the centre transform coefficient $x_{i,j}$. The vertical filter 1602

receives the groups of coefficients in a "raster" type scan order, viz the centre transform coefficient is sequentially input in raster scan order. Thus each transform coefficient $x_{i,j}$ will need to be input into the vertical filter 1602 a number of times.

The intermediate transform coefficients $y_{i,j}$, $y_{i,j+1}$ and so on are fed to the adder 1606 in a pipeline manner and added to respective intermediate transform coefficients $y'_{i,j}$, $y'_{i,j+1}$ from another vertical line filter (not shown). The added transform coefficients are then interpolated in the horizontal direction (up sampled) by an interpolator 1608. The interpolated transform coefficients $y''_{i,j}$ are then fed to a horizontal line filter 1604.

At any one clock cycle, the horizontal filter 1604 takes one interpolated coefficient $y''_{i,j}$ as input. The horizontal filter 1604 then calculates and outputs 1610 one coefficient $z_{i,j-4}$ based on the input transform coefficients { $y''_{i,j}$, $y''_{i,j-1}$, $y''_{i,j-2}$, $y''_{i,j-3}$, $y''_{i,j-4}$, $y''_{i,j-5}$, $y''_{i,j-6}$, $y''_{i,j-7}$, $y''_{i,j-8}$}, which are presently stored in the horizontal filter 1604. That is, the filter 1604 undertakes a one-dimensional inverse transform in the horizontal direction of the coefficients { $y''_{i,j}$, $y''_{i,j-1}$, $y''_{i,j-2}$, $y''_{i,j-3}$, $y''_{i,j-4}$, $y''_{i,j-5}$, $y''_{i,j-6}$, $y''_{i,j-7}$, $y''_{i,j-8}$

At the next clock cycle, the horizontal filter 1604 takes as input the next interpolated coefficient $y''_{i,j+1}$ and calculates and outputs 1610 the inverse transform coefficient based on the coefficients { $y''_{i,j+1}$, $y''_{i,j}$, $y''_{i,j-1}$, $y''_{i,j-2}$, $y''_{i,j-3}$, $y''_{i,j-4}$, $y''_{i,j-5}$, $y''_{i,j-6}$, $y''_{i,j-7}$} presently stored therein. In this way, the horizontal filter acts as a shift register arrangement. The horizontal filter 1604 continues in this manner until the end of the row i after which it continues at the next row and so on. As can be seen, the horizontal filter 1604 is effectively four data points behind the vertical filter in calculating the inverse transform coefficient. The arrangement is such that the inverse transform coefficients are calculated in raster scan order.

It is preferable that edge mirroring be used to overcome the problems in calculating the inverse transform at the edge of the sub-band. For example, at the first

clock cycle, the vertical filter 1602 takes as its input nine transform coefficients $\{x_{-4,0}, x_{-3,j}, x_{-2,0}, x_{-1,0}, x_{0,0}, x_{1,0}, x_{2,0}, x_{3,0}, x_{4,0}\}$ where the first (and last) four samples are mirrored as they are read into the vertical filter 1602. Alternatively, the first four samples may be set to zero. As mentioned above, the horizontal filter 1604 is effectively four data points behind in calculating the inverse transform coefficient. Thus, the horizontal filter can be clocked to commence at the fifth sample in each row with the first (and last) four samples mirrored.

In this way, the arrangement provides for processing two-dimensional separable convolutional kernels such that the filtered output data can be produced in a linear direction with minimal intermediate memory and at a high speed.

Turning now to Fig. 15, there is shown a block diagram of a parallel convolver suitable for use as the vertical filter 1602 shown in Fig. 14. The vertical filter 1602 comprises a memory array 1802 for receiving the nine transform coefficients $\{x_{i-4,j}, x_{i-3,j}, x_{i-2,j} x_{i-1,j}, x_{i,j}, x_{i+1,j}, x_{i+2,j}, x_{i+3,j}, x_{i+4,j}\}$ of the wavelet transform sub-band as input 1601 (see Fig. 14). The vertical filter 1602 further comprises four adders 1804, 1806, 1808, and 1810 for adding together the transform coefficients stored in the memory array 1802. In this regard, the adder 1804 adds together the transform coefficients $x_{i-4,j}$ and $x_{i+4,j}$, the adder 1806 adds together the transform coefficients $x_{i-3,j}$ and $x_{i+3,j}$, the adder 1808 adds together the transform coefficients $x_{i-2,j}$ and $x_{i+2,j}$, and the adder 1810 adds together the transform coefficients $x_{i-1,j}$ and $x_{i+1,j}$. The vertical filter 1602, in addition, comprises five memory stores 1812, 1814, 1816, 1818 and 1820 for storing the filter coefficients h_8, h_7, h_6, h_5, and h_4. The vertical filter also comprises four multipliers 1822, 1824, 1826, and 1828 for multiplying the added transform coefficients output by the adders 1804, 1806, 1808, and 1810 by the filter coefficients h_8, h_7, h_6, and h_5 stored in memory stores 1812, 1814, 1816, and 1818 respectively. In addition, the vertical filter 1602 comprises a

further multiplier 1830 for multiplying the centre transform coefficient $x_{i,j}$ stored in memory array 1802 by the centre filter coefficient h_4 stored in memory 1820. The results of the multipliers 1822, 1824, 1826, 1828, and 1830 are then fed to a summer 1832 which adds the results of the multipliers and feds them to an output channel 1605 of the vertical filter 1602. In this way, the vertical filter 1602 outputs an intermediate transform coefficient $y_{i,j}$ one per clock cycle.

The typical representation of a convolution computation is expressed mathematically as:

$$y(n) = \sum_{k} h(n-k)\, x\, (k) \qquad (1)$$

where y(n) = nth output point

h(n-k) = kth filter tap coefficient

x(k) = kth input point

The parallel convolver 1602 shown in Fig. 15 takes advantage of the symmetry of the inverse wavelet kernel in that it minimizes the number of multipliers. The convolver 1602 adds the mirror image counterpart of a transform coefficient around the centre filter coefficient of the filter before applying the multiplication by the filter. Such may be derived by re-arrangement of the convolution computation shown in Eqn (1) as follows:

$$y_{i,j} = h_0 x_{i-c,j} + h_1 x_{i-c+1,j} + \ldots\ldots h_{2c} x_{i+c,j} \qquad \text{Eqn. (2)}$$

where: c = (N-1)/2 and N is odd, (in this particular example, N = 9).

As the wavelet is symmetrical then: $h_{c+k} = h_{c-k}$ .....$(k < c)$ thereby enabling re-arrangement of the convolution computation (Eqn. (2)) to:

$$y_{i,j} = h_c x_{1,j} + h_{c+1}(x_{i+1,j} + x_{i-1,j}) + h_{c+2}(x_{i+2,j} + x_{i-2,j}) + \ldots h_{2c}(x_{i+c,j} + x_{i-c,j}) \qquad \text{Eqn(3)}$$

Turning now to Fig. 16, there is shown a block diagram of a sequential convolver suitable for use as the horizontal filter 1604 shown in Fig. 14. The structure of the horizontal filter 1604 is substantially the same as the vertical filter 1602, with the exception of the input arrangement of the filter and the values of the filter coefficients h_i. As mentioned previously, the immediate transform coefficients are input into the horizontal filter 1604, one at a time, which acts in a shift register manner. The horizontal filter 1604 comprises a memory array 1902 for receiving the nine immediate transform coefficients $\{ y''_{i,j}, y''_{i,j-1}, y''_{i,j-2}, y''_{i,j-3}, y''_{i,j-4}, y''_{i,j-5}, y''_{i,j-6}, y''_{i,j-7}, y''_{i,j-8}\}$ in a sequential manner. The horizontal filter 1604 further comprises four adders 1904, 1906, 1908, and 1910 for adding together the transform coefficients stored in the memory array 1902. Namely, adder 1904 adds together the transform coefficients $y''_{i,j}$ and $y''_{i,j-8}$, the adder 1906 adds together the transform coefficients $y''_{i,j-1}$ and $y''_{i,j-7}$, the adder 1908 adds together the transform coefficients $y''_{i,j-2}$ and $y''_{i,j-6}$, and the adder 1910 adds together the transform coefficients $y''_{i,j-3}$ and $y''_{i,j-5}$. The horizontal filter 1604, in addition, comprises five memory stores 1912, 1914, 1916, 1918 and 1920 for storing the filter coefficients h_8, h_7, h_6, h_5, and h_4. The horizontal filter 1604 also comprises four multipliers 1922, 1924, 1926, and 1928 for multiplying the added transform coefficients output by the adders 1904, 1906, 1908, and 1910 by the filter coefficients h_8, h_7, h_6, and h_5 stored in memory stores 1912, 1914, 1916, and 1918 respectively. In addition, the horizontal filter 1604 comprises a further multiplier 1930 for multiplying the centre transform coefficient $y''_{i+4,j}$ stored in memory array 1902 by the centre filter coefficient h_4 stored in memory 1920. The results of the multipliers 1922, 1924, 1926, 1928, and 1930 are then fed to a summer 1932 which adds the results of the multipliers and feds them to an output channel 1610 of

the horizontal filter 1604. In this way, the horizontal filter 1604 outputs an inverse transform coefficient one per clock cycle.

Turning now to Fig. 17, there is shown an arrangement 2000 of part of an inverse two-dimensional separable DWT transformer which is similar to the arrangement shown in Fig. 14, except that the arrangement 2000 has the advantage of accessing multiple lines of the sub-band and output multiple coefficients simultaneously. This advantage is achieved by multiple parallel convolvers (2001-1,2001-2) and sequential convolvers (2104-1,2104-2) (see Fig. 18) arranged in parallel. Namely, the arrangement 2000 comprises multiple vertical filters (2001-1, 2001-2) with a common memory array 2002 and corresponding multiple horizontal filters (2104-1,2104-2) in parallel.

For ease of explanation, only two parallel vertical and horizontal filter paths are shown, although the embodiment is not intended to be limited thereto. The arrangement 2000 can comprise many more parallel vertical and horizontal filter paths. Furthermore, the vertical and horizontal filters filter 1602 and 1604 (see Fig. 14) both have nine taps but again it is not intended to be limited thereto.

The arrangement 2000 comprises two vertical filters 2001-1 and 2001-2 having a common memory array 2002. The memory array 2002 has parallel inputs for receiving ten transform coefficients $\{ x_{i-5,j} , x_{i-4,j} , x_{i-3,j} , x_{i-2,j} x_{i-1,j} , x_{i,j} , x_{i+1,j} , x_{i+3,j}, x_{i+3,j} , x_{i+4,j} \}$ of the wavelet transform sub-band as input. The nine transform coefficients $\{x_{i-4,j} , x_{i-3,j} , x_{i-2,j} x_{i-1,j} , x_{i,j} , x_{i+1,j} , x_{i+2,j} , x_{i+3,j} , x_{i+4,j} \}$ are then input to the vertical filter 2001-1 and the nine transform coefficients $\{ x_{i-5,j} , x_{i-4,j} , x_{i-3,j} , x_{i-2,j} x_{i-1,j} , x_{i,j} , x_{i+1,j} , x_{i+2,j} , x_{i+3,j} \}$ are then input to the vertical filter 2001-2. Namely, the coefficient $x_{i+4,j}$ is input via 1 to adder 2004-1, the coefficient $x_{i+3,j}$ is input via 2 to adders 2006-1 and 2004-2, the coefficient $x_{i+2,j}$ is input via 3 to adders 2008-1 and 2006-2, and so on. In this way, the vertical filters

2001-1 and 2001-2 simultaneously calculate the intermediate transform coefficients for the locations i,j and i-1,j respectively.

The vertical filters 2001-1 and 2001-2 are both substantially the same and only filter 2001-1 will be described in detail. The vertical filter 2001-1 further comprises four adders 2004-1, 2006-1, 2008-1, and 2010-1 for adding together the transform coefficients stored in the memory array 2002. Namely, adder 2004-1 adds together the transform coefficients $x_{i-4,j}$ and $x_{i+4,j}$, the adder 2006-1 adds together the transform coefficients $x_{i-3,j}$ and $x_{i+3,j}$, the adder 2008-1 adds together the transform coefficients $x_{i-2,j}$ and $x_{i+2,j}$, and the adder 2010-1 adds together the transform coefficients $x_{i-1,j}$ and $x_{i+1,j}$. The vertical filter 2001-1, in addition, comprises five memory stores 2012-1, 2014-1, 2016-1, 2018-1 and 2020-1 for storing the filter coefficients h_8, h_7, h_6, h_5, and h_4. The vertical filter also comprises four multipliers 2022-1, 2024-1, 2026-1, and 2028-1 for multiplying the added transform coefficients output by the adders 2004-1, 2006-1, 2008-1, and 2010-1 by the filter coefficients h_8, h_7, h_6, and h_5 stored in memory stores 2012-1, 2014-1, 2016-1, and 2018-1 respectively. In addition, the vertical filter 2001-1 comprises a further multiplier 2030-1 for multiplying the centre transform coefficient $x_{i,j}$ stored in memory array 2002 by the centre filter coefficient h_4 stored in memory 2020-1. The results of the multipliers 2022-1, 2024-1, 2026-1, 2028-1, and 2030-1 are then fed to a summer 2032-1 which adds the results of the multipliers and feeds them to an output channel 2005-1 of the vertical filter 2001-1. Thus, the vertical filters 2001-1 and 2001-2 outputs intermediate transform coefficients $y_{i,j}$ and $y_{i-1,j}$ respectively one per clock cycle.

The vertical filters 2001-1 and 2001-2 are coupled to the horizontal filters 2104-1 and 2104-2 respectively via adders 2140 and 2142 and interpolators 2144 and 2146 (see Fig. 18). The intermediate transform coefficients $y_{i,j}$, $y_{i-1,j}$ are fed to respective adders 2140 and 2142 in a pipeline manner and added to intermediate transform coefficients $y'_{i,j}$

$y'_{i-1,j}$ from other vertical line filters (not shown). The added transform coefficients are then interpolated in the horizontal direction (up sampled) by interpolators 2144 and 2146.

The interpolated transform coefficients $y''_{i,j}$, $y''_{i-1,j}$ are then fed to respective horizontal line filters 2104-1 and 2104-2. The horizontal line filters 2104-1 and 2104-2 each comprise a sequential convolver of the type described with reference to Fig. 16. The horizontal filters 2104-1 and 2104-2 output the inverse transformed image coefficients 2110-1, 2110-2 in a pipeline manner. The arrangement is such that the inverse transform coefficients 2110-1, 211002 are calculated two rows at a time in raster order.

A number of different implementations aimed at achieving hardware cost reductions through re-use of hardware modules is now described. This re-use is made possible because of the up-sampling inherent between successive levels of the IDWT. The consequence of this is that computational blocks associated with the final inverse transform level run at the highest speed, with computational blocks associated with previous levels having progressively more "idle time".

Fig. 19 makes use of this idle time, the architecture described in relation to Fig. 13 being replicated for each level, with a computation circuit 1408 at level j feeding the LL data 1410 into an identical computational circuit 1406 at level j-1, and so on in a cascade arrangement. Dashed blocks have been used to represent the computational blocks 1408 and 1406 to indicate that they are virtual computational blocks, resulting from time sharing the "real" computational block 1402 at the various levels. The set of computational blocks for all levels 1 through j comprises a computational pipeline.

In order to produce a single output point at a given level j-1, assuming a low pass filter of 9 taps, a minimum 5 × 5 points are required for each sub-band at the preceeding level j, which are then up-sampled before filtering. To ensure uniformity of implementation across levels, as well as stable filter sizes as the filters are "slid", 8 × 8

points are defined as being required from each sub-band in the preceeding levels. As illustrated in Fig. 6 for the one-dimensional case, this implies that all 8 × 8 points from each sub-band must be available from the preceeding level j, before any operations can occur at level j-1. This implies, that in order to fill the pipeline, LL sub-band data must be generated from level j, then for level j-1, and thereafter for every level up to level 1, so that the complete computational pipeline is filled.

With reference to Fig. 13, this process occurs sequentially, as LL data is generated by stepping a column of data eg. 1302 from each of the input sub-bands through the vertical parallel convolvers eg. 1304, which then feed the sequential convolvers eg. 1316 on each cycle, after the requisite summations are performed eg. in adder 1332. Data is clocked into the circuit on each cycle until seven sets of seven points 1324 are generated as output from the serial convolvers eg. 1316 being added in adders 1338. This LL data is generated one column at a time, and the generation of each column signals the computational circuit at the next level to accept this, and also to retrieve the LH, HL and HH columns associated therewith, to perform a vertical parallel convolution step. Once a level has generated seven columns of output LL data 1324, further columns of input data for each of the sub-bands is only clocked into the parallel convolvers, eg. 1304 when the next level requires an additional column of data. After the pipeline is filled, each level operates at half the rate of the next level, ie. level j operates at half the rate of level j-1.

At level 1, the computational circuit is capable of producing one column of output data, eg. 1324, on each clock cycle. However, for every column of sub-band data that is input into this level, two columns are generated at the output, due to the up-sampling of input data within the convolvers. Therefore, level 1 requires input data in the form of columns from the LL, LH, HL and HH sub-bands at half the rate that it produces

output. Again, as described in relation to the one dimensional case, since this occurs at all levels, it is apparent that one set of four vertical parallel convolvers can satisfy the requirements of all levels beyond level 1, ie. can satisfy the requirements of level 2 through to level j.

Because no state information is required in the vertical parallel convolvers eg. 1304, one set of four vertical parallel convolvers can be shared amongst levels 2 to j. A multiplexer at the input of each convolver feeds sub-band data from the required level at the appropriate time.

This is illustrated in Fig. 20, where sub-band data 1500 which is seven points wide is fed into a multiplexer 1502, thereafter being selectively fed to a (shared) vertical convolver 1504. This convolver 1504 produces seven outputs points 1510 which are directed to an output multiplexer 1506, which in turn directs the sub-band data 1508 to the appropriate horizontal convolver. Output from the parallel convolvers are fed back to the level that is supplying the input at the current time.

The same approach can be applied to the serial convolvers eg. 1316 associated with each level. It is noted however that the serial convolvers contain state information which must be preserved, and therefore if this approach is adopted, a context store eg. 1404 (see Fig. 19) is provided in relation to the horizontal convolver, which stores state information for each serial convolver eg. 1316 for a particular level, while the serial convolver is being used at another level.

The storage of horizontal sequential convolver state further enables a multi-stage implementation of the computational pipeline. This technique is useful if it is determined that hardware cost is a limiting factor, and yet that further layers of tranformation are required. We recall in relation to Fig. 19 how a real computational block 1402 is shared to form a virtual computational block eg 1406 during the idle time associated with level j-2.

The multi-stage embodiment takes this use of virtual computational blocks one step further, as shown in Fig. 21. The storage of context to enable multiple stages of computational pipeline may be effected by a register store for each sequential convolver in the computational pipeline. This is shown in the Fig 22, in which a context store 2308 is associated with each horizontal convolver 2306. One such store 2308 is requested per horizontal convolver 2306 per pipeline stage. Thus, if a two stage pipeline is required, two stores 2308 are required for each horizontal convolver 2306. This enables hardware associated with level 1 ie 2300 to be logically mapped to other levels 2302, 2304 at the cost of additional memory for the context stores. This arrangement provides a context store for the complete stage of the computational pipeline. The number of stages which can be supported is determined by the number of context stores 2308 associated with each and every sequential convolver 2306 in the design.

The method of the IDWT is preferably practiced using a conventional general-purpose computer system 1700, such as that shown in Fig. 23 wherein the process of the IDWT can be implemented as software, such as an application program executing within the computer system 1700. In particular, the steps of the method of the IDWT are effected by instructions in the software that are carried out by the computer. The software can be divided into two separate parts; one part for carrying out the IDWT methods; and another part to manage the user interface between the latter and the user. The software can be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer from the computer readable medium, and then executed by the computer. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer preferably effects an

advantageous apparatus for performing an IDWT in accordance with the embodiments of the invention.

The computer system 1700 comprises a computer module 1701, input devices such as a keyboard 1702 and mouse 1703, output devices including a printer 1715 and a display device 1714. A Modulator-Demodulator (Modem) transceiver device 1716 is used by the computer module 1701 for communicating to and from a communications network 1720, for example connectable via a telephone line 1721 or other functional medium. The modem 1716 can be used to obtain access to the Internet, and other network systems, such as a Local Area Network (LAN) or a Wide Area Network (WAN).

The computer module 1701 typically includes at least one processor unit 1705, a memory unit 1706, for example formed from semiconductor random access memory (RAM) and read only memory (ROM), input/output (I/O) interfaces including a video interface 1707, and an I/O interface 1713 for the keyboard 1702 and mouse 1703 and optionally a joystick (not illustrated), and an interface 1708 for the modem 1716. A storage device 1709 is provided and typically includes a hard disk drive 1710 and a floppy disk drive 1711. A magnetic tape drive (not illustrated) may also be used. A CD-ROM drive 1712 is typically provided as a non-volatile source of data. The components 1705 to 1713 of the computer module 1701, typically communicate via an interconnected bus 1704 and in a manner which results in a conventional mode of operation of the computer system 1700 known to those in the relevant art. Examples of computers on which the embodiments can be practised include IBM-PC's and compatibles, Sun Sparcstations or alike computer systems evolved therefrom.

Typically, the application program of the preferred embodiment is resident on the hard disk drive 1710 and is read and controlled in its execution by the processor 1705. Intermediate storage of the program and any data fetched from the network 1720 may be

accomplished using the semiconductor memory 1706, possibly in concert with the hard disk drive 1710. In some instances, the application program may be supplied to the user encoded on a CD-ROM or floppy disk and read via the corresponding drive 1712 or 1711, or alternatively may be read by the user from the network 1720 via the modem device 1716. Still further, the software can also be loaded into the computer system 1700 from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer module 1701 and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets including email transmissions and information recorded on websites and the like. The foregoing is merely exemplary of relevant computer readable mediums. Other computer readable mediums may be practiced without departing from the scope and spirit of the invention.

The method of the IDWT can alternatively be implemented in dedicated hardware such as one or more integrated circuits performing the functions or sub functions of the IDWT. Such dedicated hardware may include digital signal processors, or one or more microprocessors and associated memories.

Industrial Applicability

It is apparent from the above that the embodiments of the invention are applicable to the computer and data processing industries.

The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiments being illustrative and not restrictive.

Claims:

1. A method for performing an Inverse Discrete Wavelet Transform (IDWT) comprising, for a first sub-band level and a subsequent sub-band level in an N level Discrete Wavelet Transform, the steps of:

(i) inverse transforming, using filters having associated filter widths, data from associated sub-bands in the first sub-band level, to form processed data in a corresponding sub-band in the subsequent sub-band level; and

(ii) inverse transforming, using subsequent filters having the same corresponding associated filter widths, the processed data in conjunction with corresponding data from associated sub-bands in the subsequent sub-band level.

2. A method according to claim 1, whereby steps (i) and (ii) are performed in a parallel cascade for all N-1 pairs of consecutive levels, this being performed directly and substantially without intermediate memory.

3. A method according to claim 1, whereby first filters are used in relation to level 1, and second filters are time shared by all other N-1 pairs of consecutive levels 2 and 3, 3 and 4, .. N-1 and N, the second filters being substantially applied to only a single pair of levels at a given time.

4. A method according to claim 3, whereby time sharing is performed using a time multiplexer which multiplexes data from pairs of levels to the second filters.

5. A method according to claim 3, where data associated with a pair of sub-band levels associated with the second filters, is stored while the second filters are being applied to another pair of sub-band levels.

6. An apparatus for for performing an Inverse Discrete Wavelet Transform (IDWT) in accordance with any one of the aforementioned methods.

7. A computer program product including a computer readable medium having recorded thereon a computer program for performing an Inverse Discrete Wavelet Transform (IDWT) in accordance with any one of the aforementioned methods.

DATED this 8th Day of December 1999

Canon Kabushiki Kaisha

Patent Attorneys for the Applicant

SPRUSON & FERGUSON

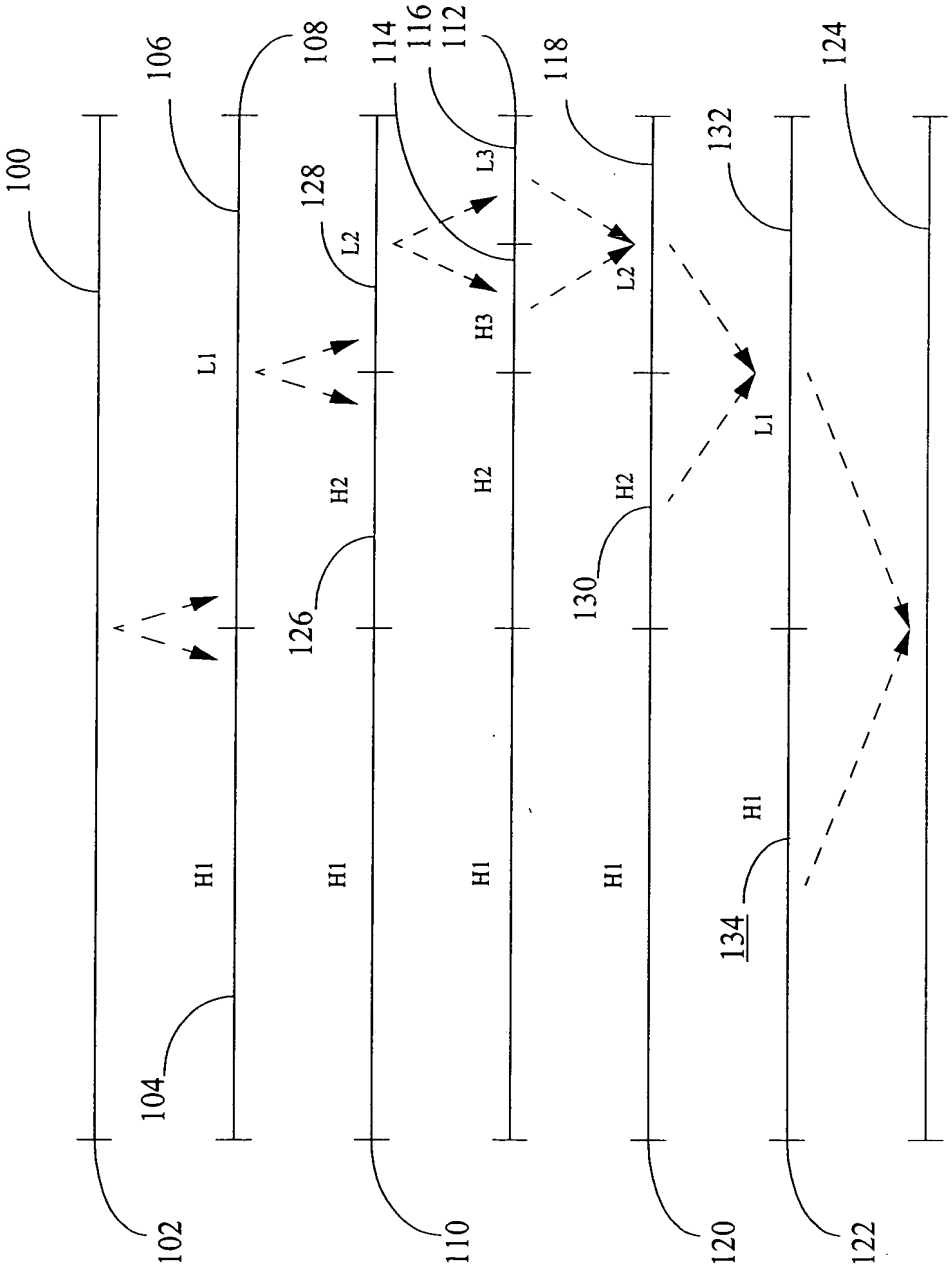

100
102
104
106
108
110
112
114
116
118
120
122
124
126
128
130
132
134
L1
L2
L3
H1
H2
H3

Fig. 1

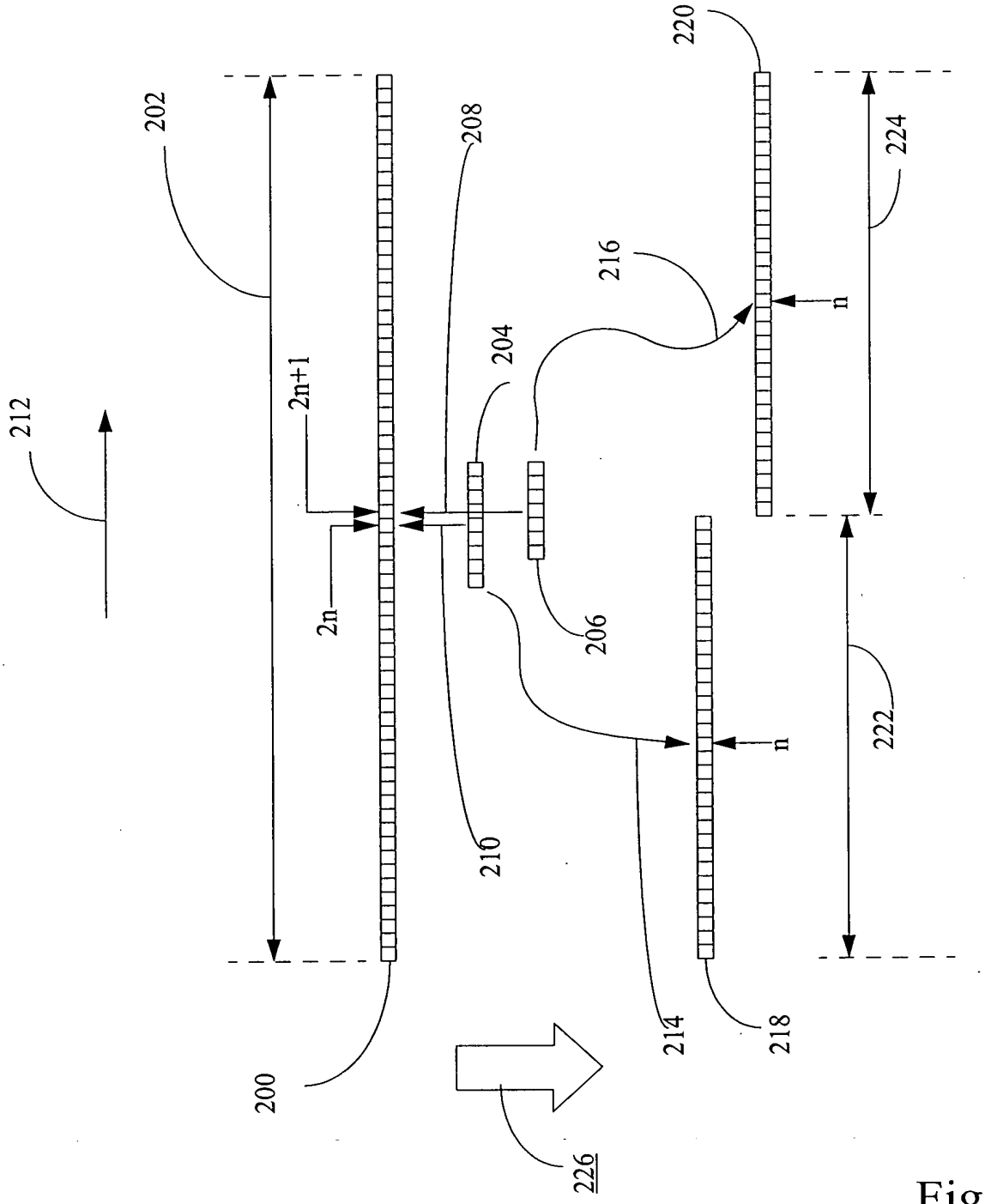


Fig. 2

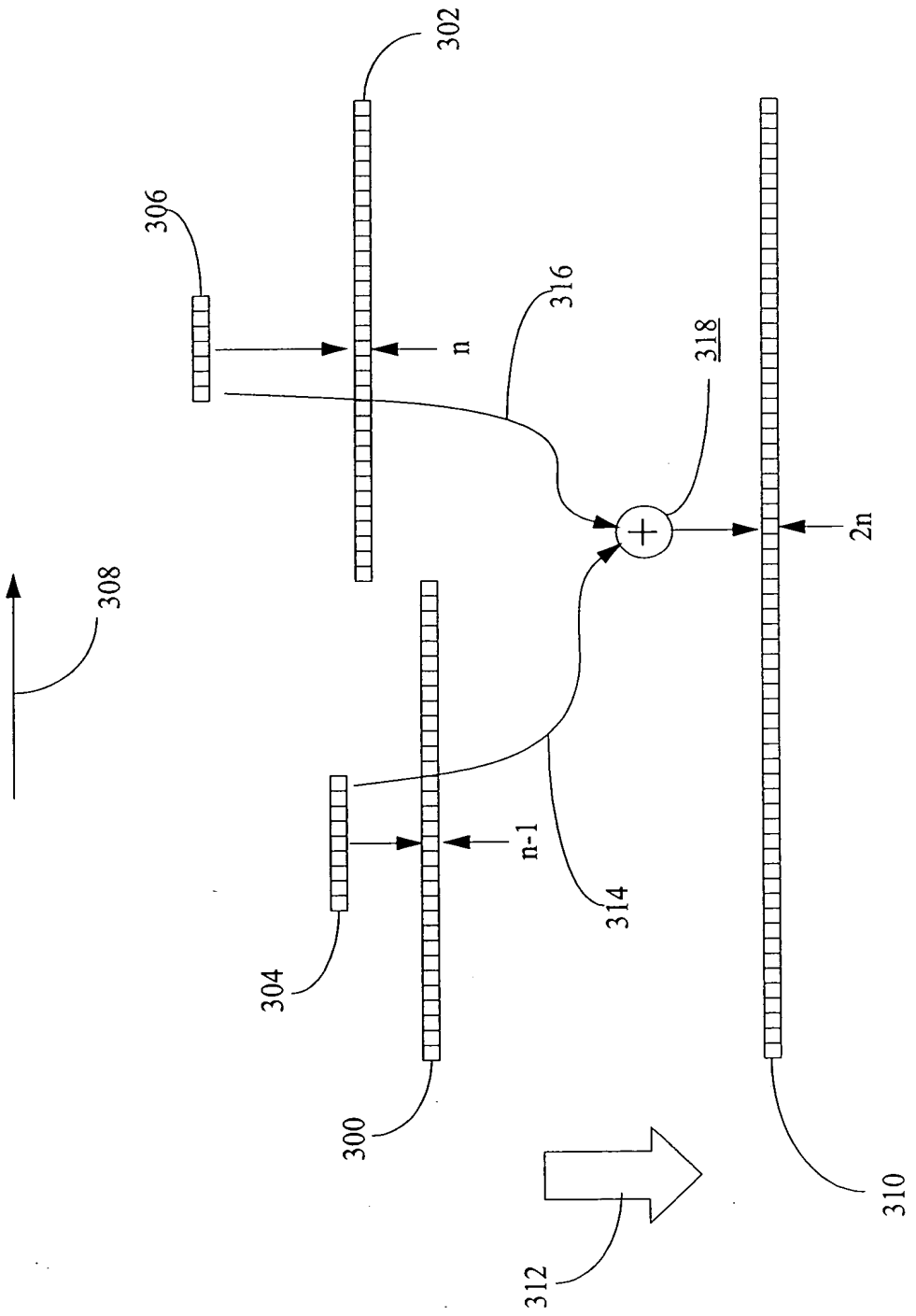


Fig. 3

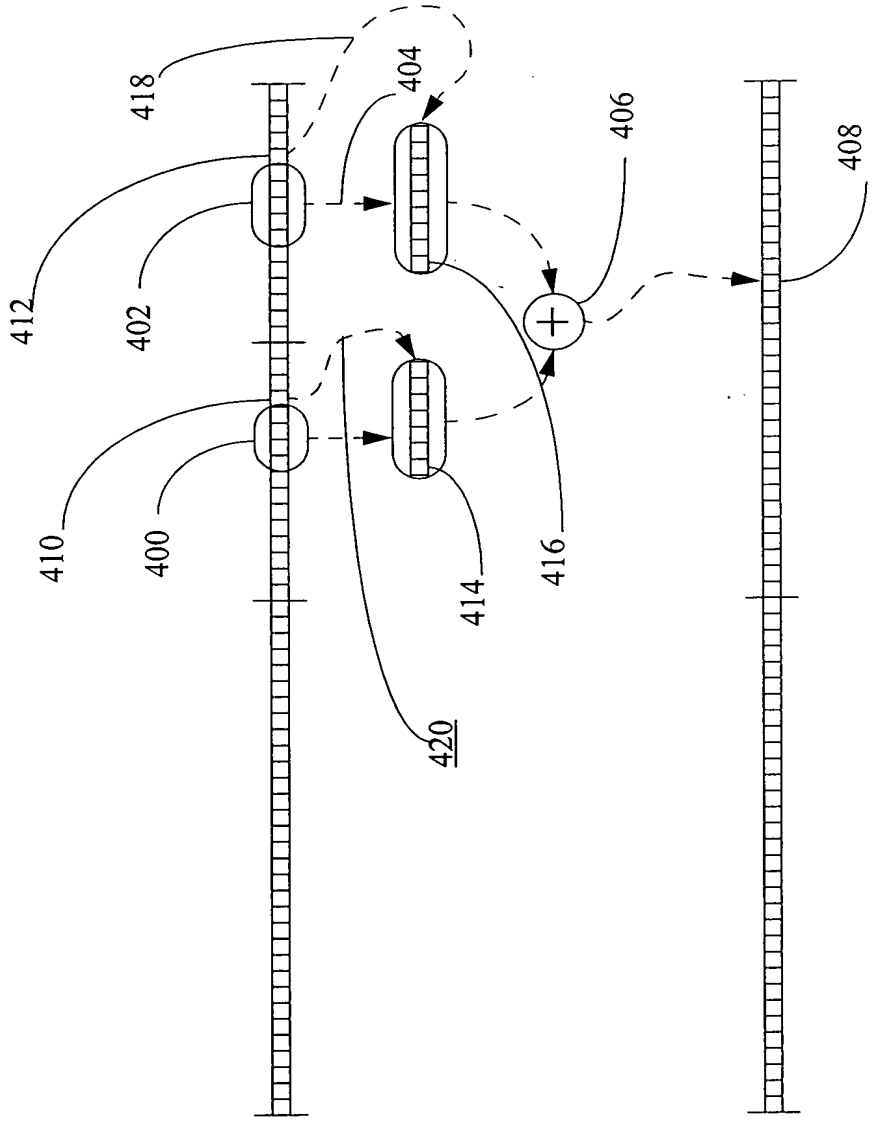


Fig. 4

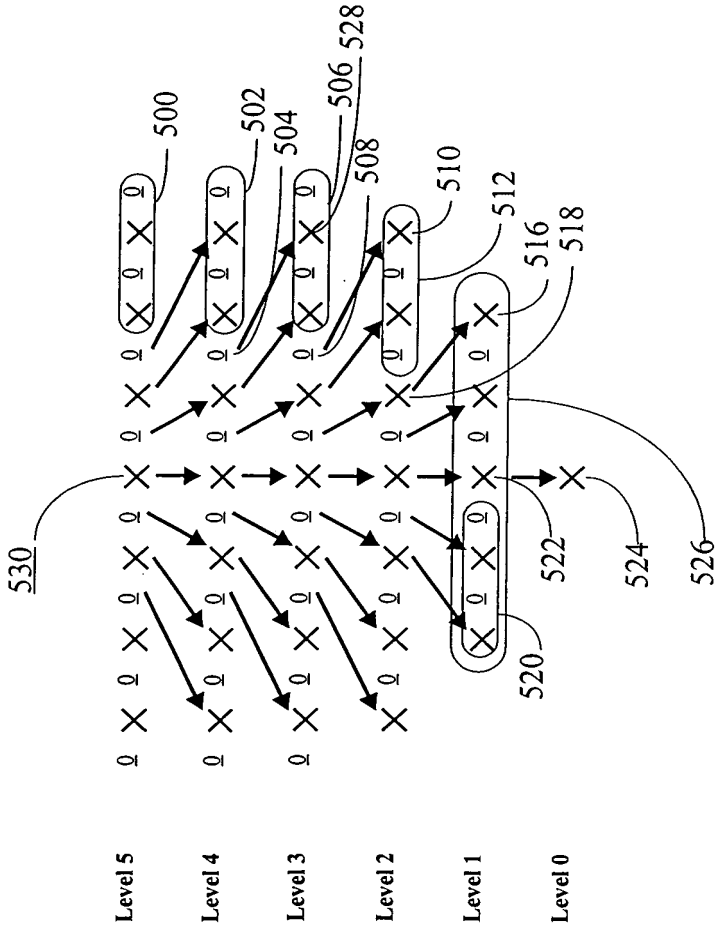

500
502
504
506
528
508
510
512
516
518
530
522
524
526
520
Level 5
Level 4
Level 3
Level 2
Level 1
Level 0

Fig. 5

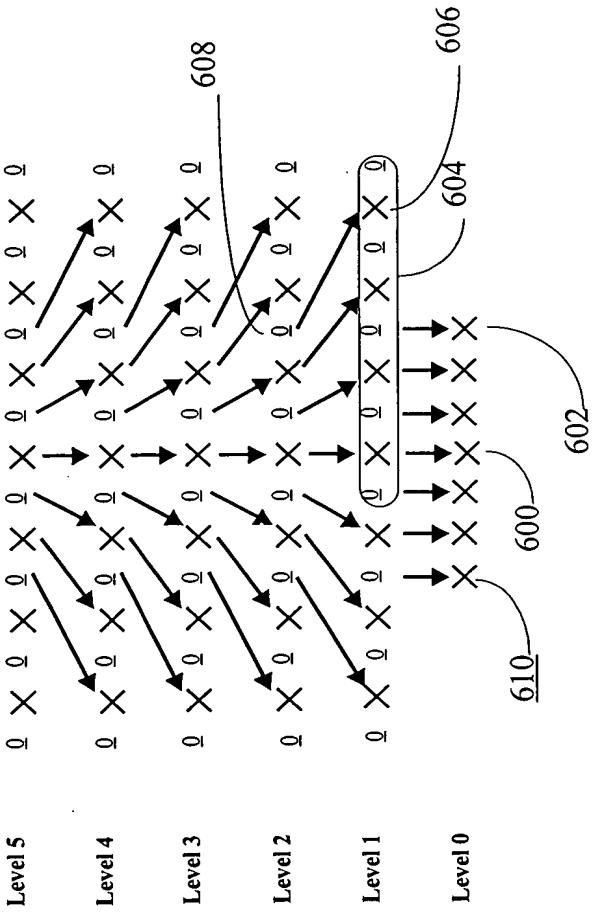

606
608
604
602
600
610
Level 5
Level 4
Level 3
Level 2
Level 1
Level 0

Fig. 6

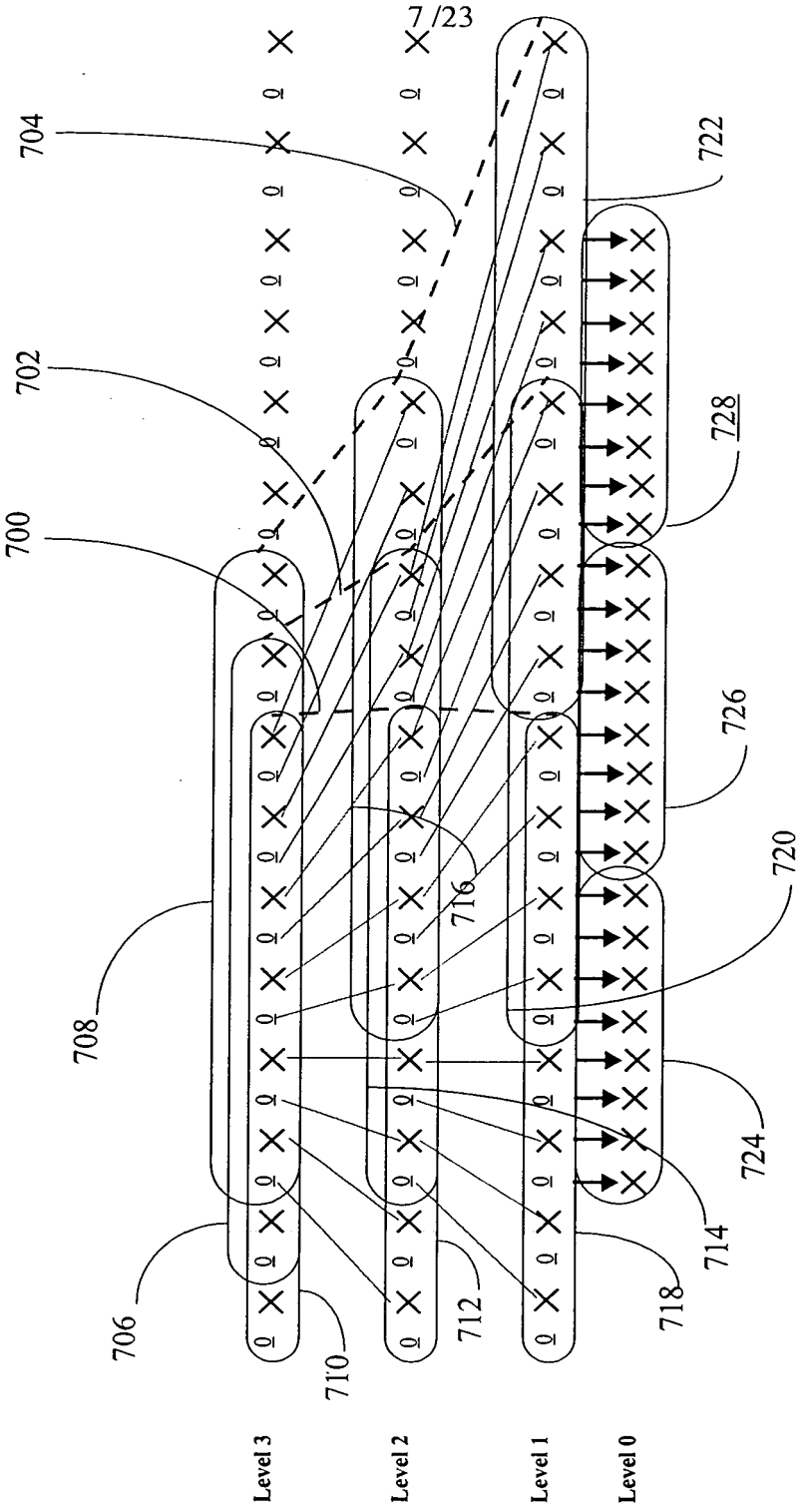

704
722
702
728
700
726
720
716
708
724
714
706
712
718
710
Level 3
Level 2
Level 1
Level 0

Fig. 7

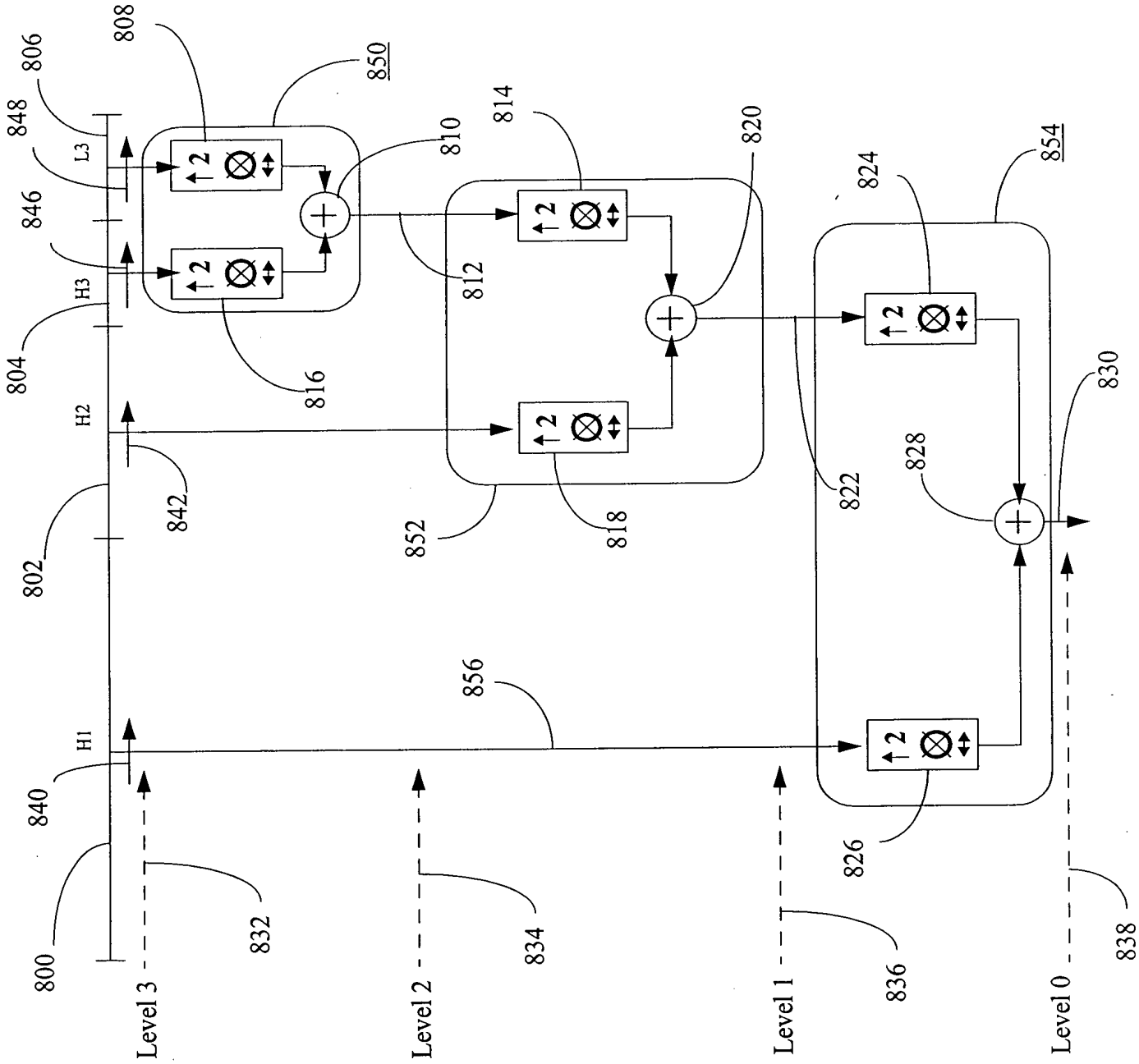

800
802
804
806
808
810
812
814
816
818
820
822
824
826
828
830
832
834
836
838
840
842
846
848
850
852
854
856
H1
H2
H3
L3
Level 3
Level 2
Level 1
Level 0

Fig. 8

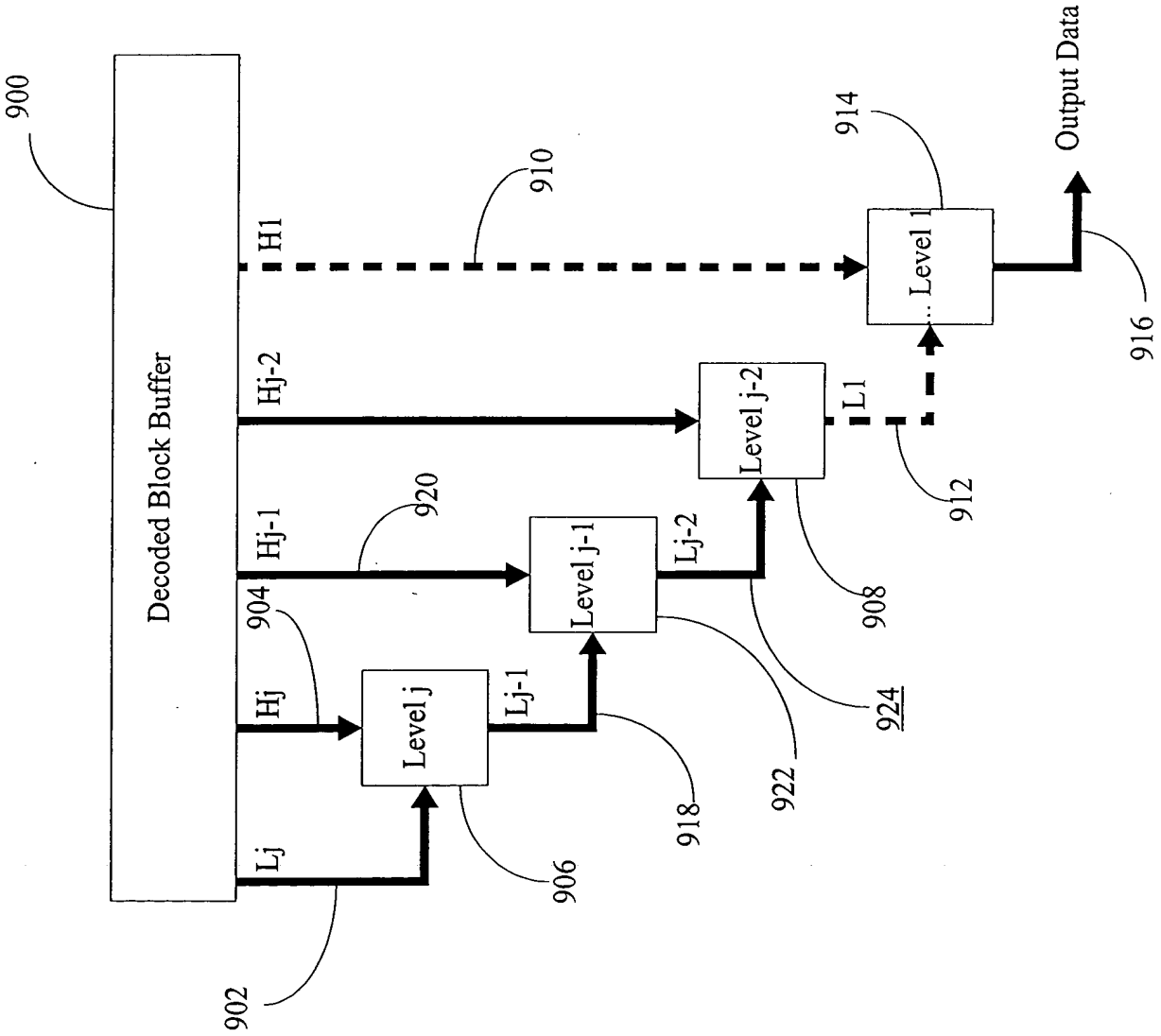

900
Decoded Block Buffer
H1
910
914
Level 1
Output Data
916
Hj-2
Level j-2
L1
912
Hj-1
920
Level j-1
Lj-2
908
904
Hj
Level j
Lj-1
924
922
918
Lj
906
902

Fig. 9

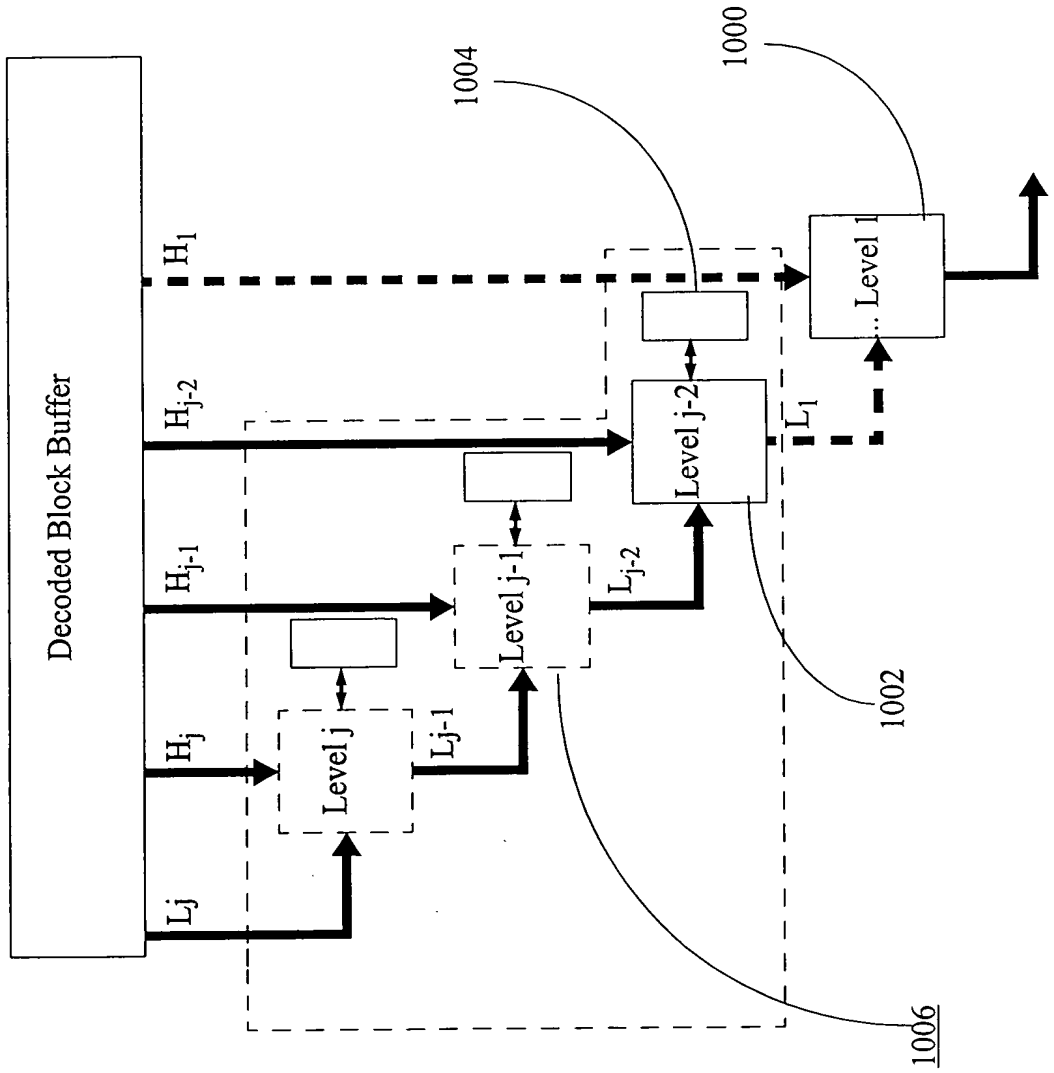

Decoded Block Buffer
H_1
H_{j-2}
H_{j-1}
H_j
L_j
Level j
L_{j-1}
Level j-1
L_{j-2}
Level j-2
L_1
... Level 1
1000
1004
1002
1006

Fig. 10

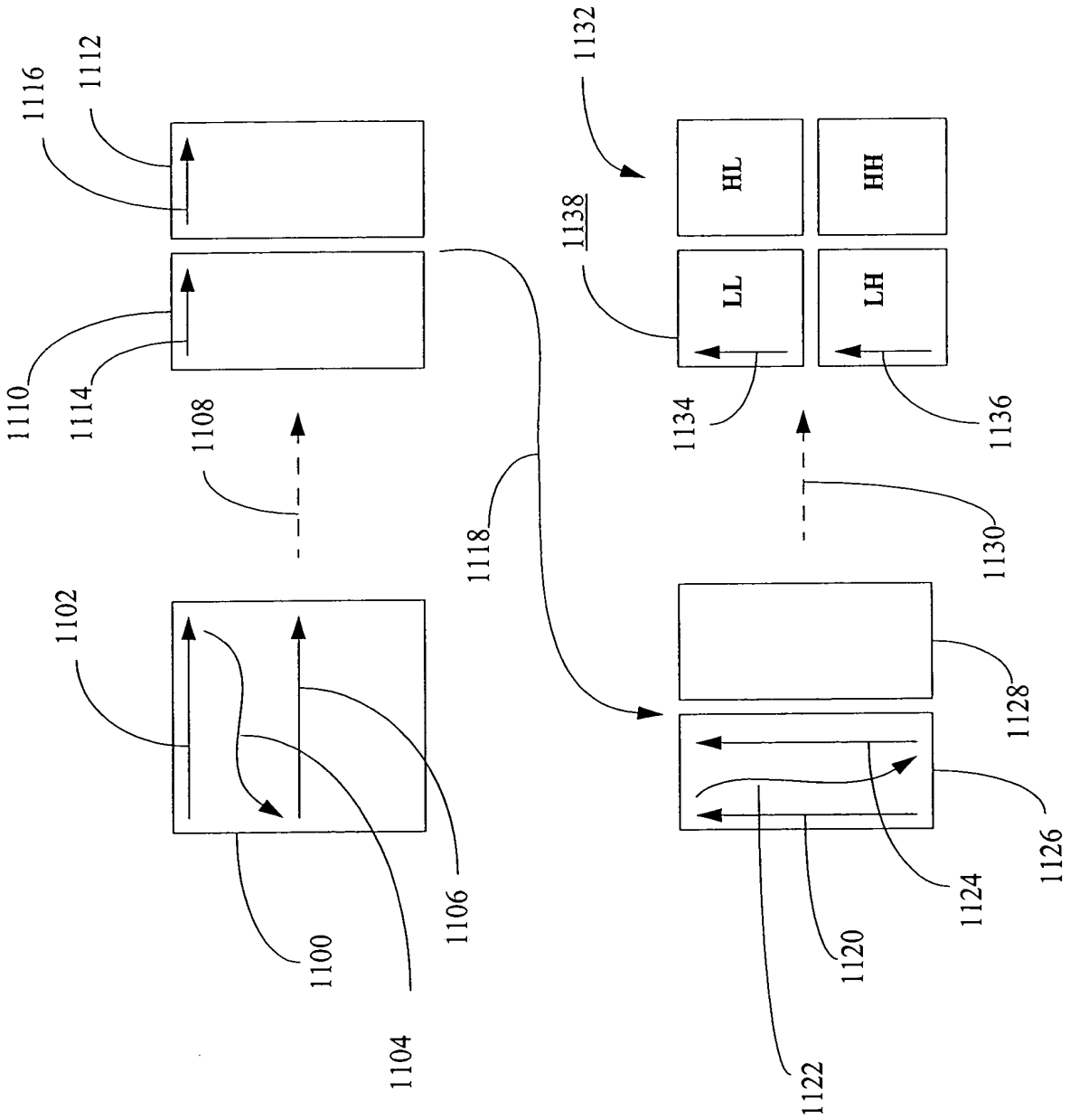
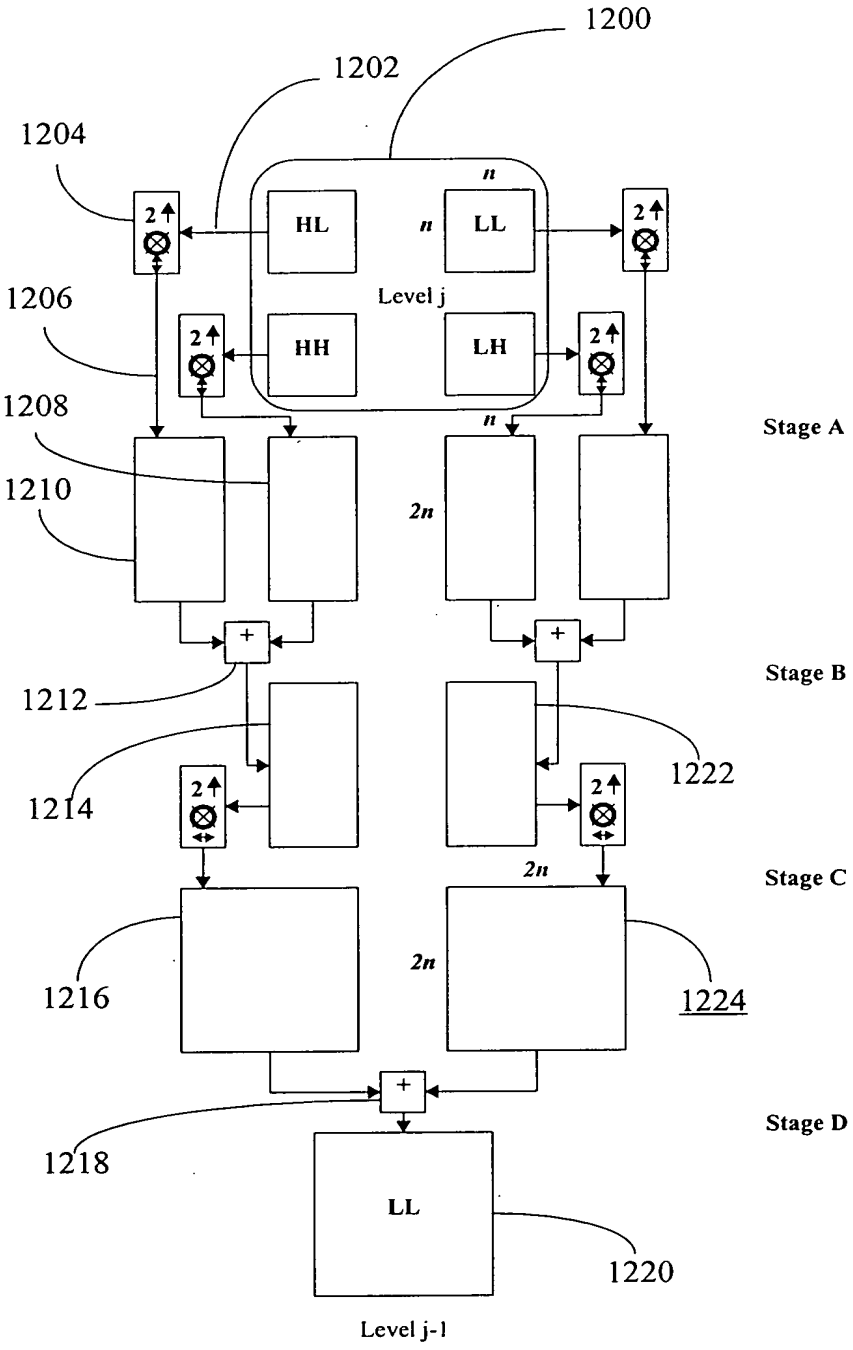

1116
1112
1110
1114
1108
1102
1100
1104
1106
1118
1132
1138
HL
HH
LL
LH
1134
1136
1130
1128
1126
1124
1120
1122

Fig. 11



(Prior art)
Fig. 12

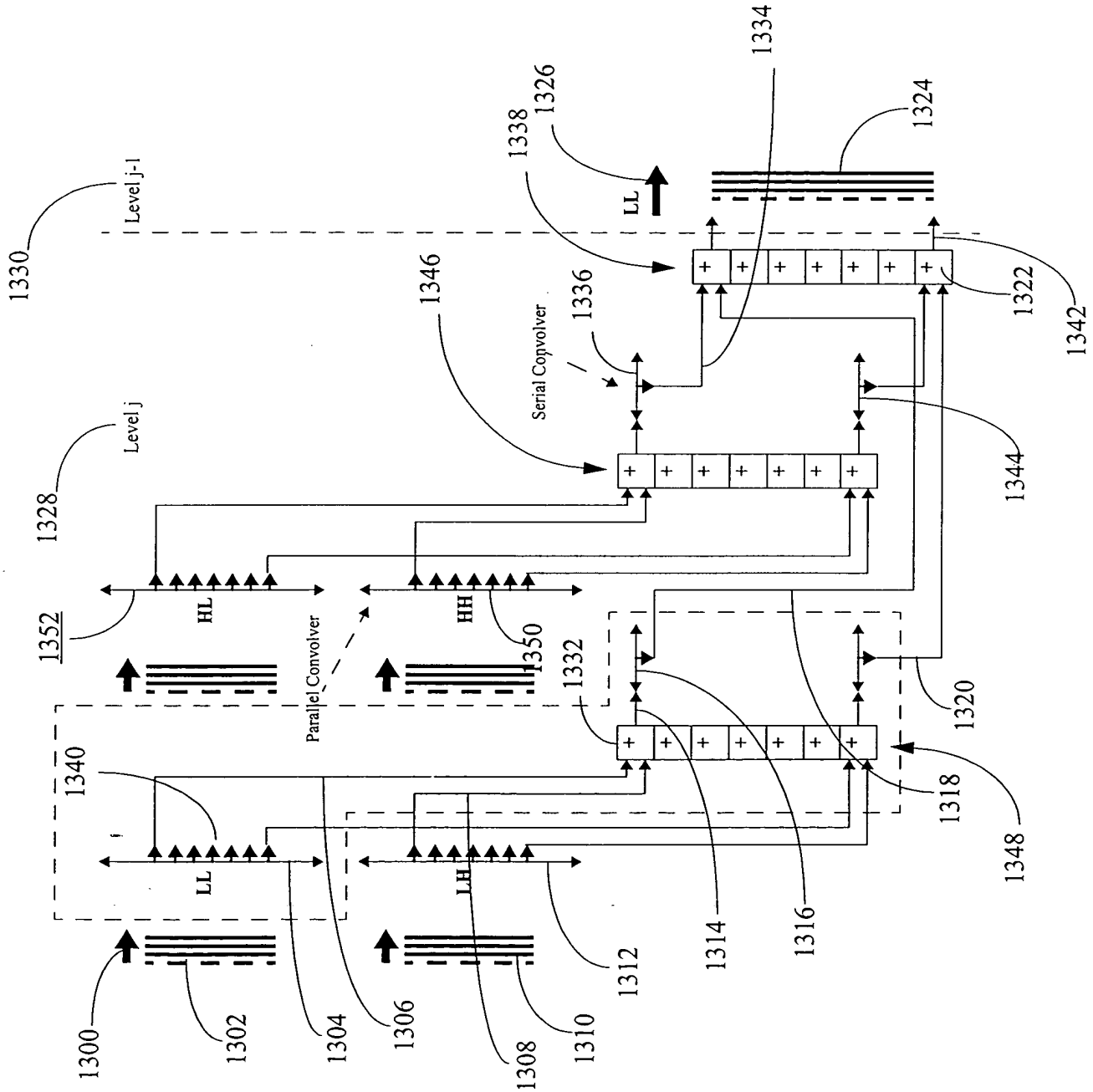

Level j-1
Level j
Serial Convolver
Parallel Convolver
LL
HL
LH
HH
1300
1302
1304
1306
1308
1310
1312
1314
1316
1318
1320
1322
1324
1326
1328
1330
1332
1334
1336
1338
1340
1342
1344
1346
1348
1350
1352

Fig. 13

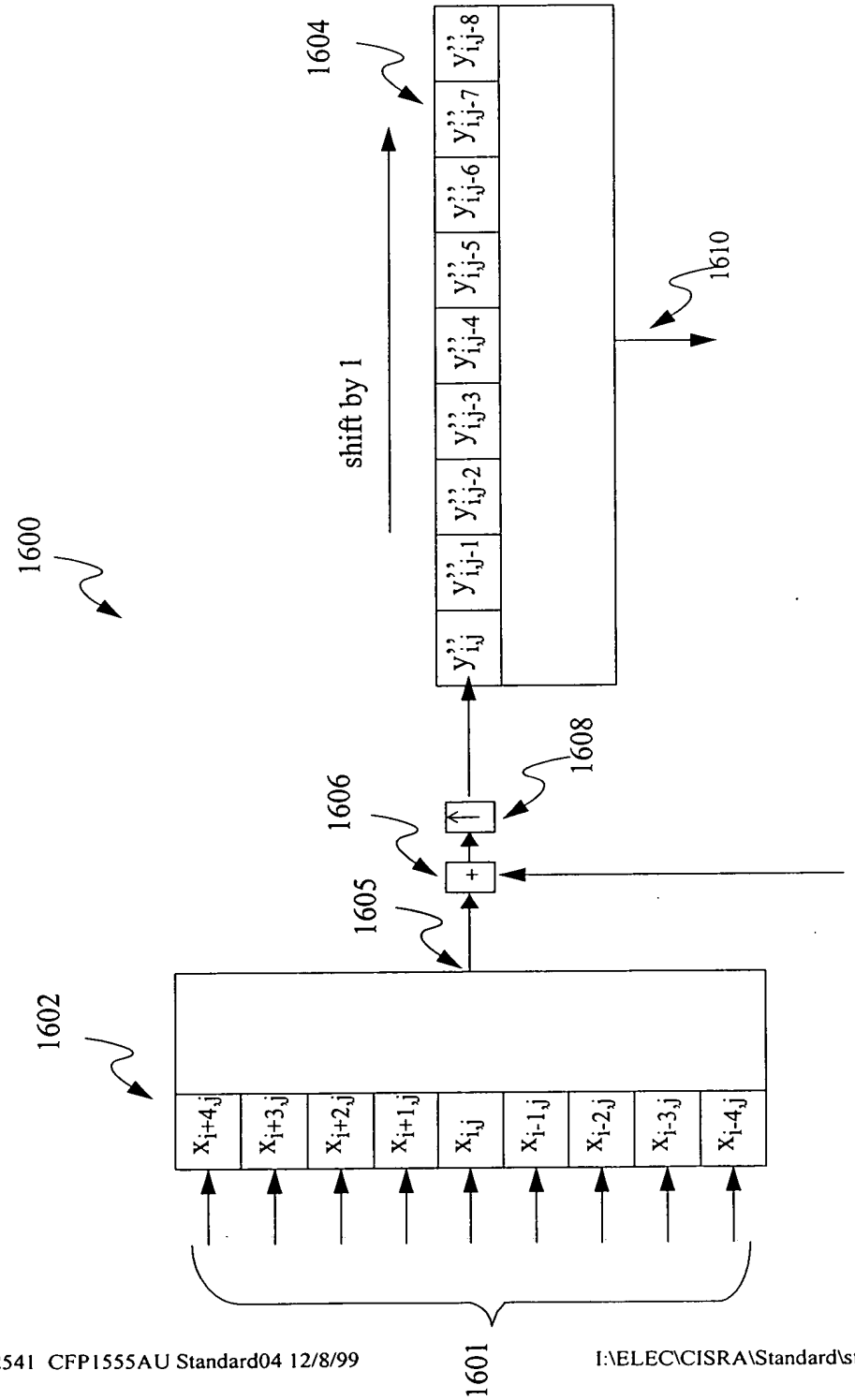

1600
1602
1604
1605
1606
1608
1610
1601
shift by 1
+
$x_{i+4,j}$
$x_{i+3,j}$
$x_{i+2,j}$
$x_{i+1,j}$
$x_{i,j}$
$x_{i-1,j}$
$x_{i-2,j}$
$x_{i-3,j}$
$x_{i-4,j}$
$y''_{i,j}$
$y''_{i,j-1}$
$y''_{i,j-2}$
$y''_{i,j-3}$
$y''_{i,j-4}$
$y''_{i,j-5}$
$y''_{i,j-6}$
$y''_{i,j-7}$
$y''_{i,j-8}$

Fig. 14

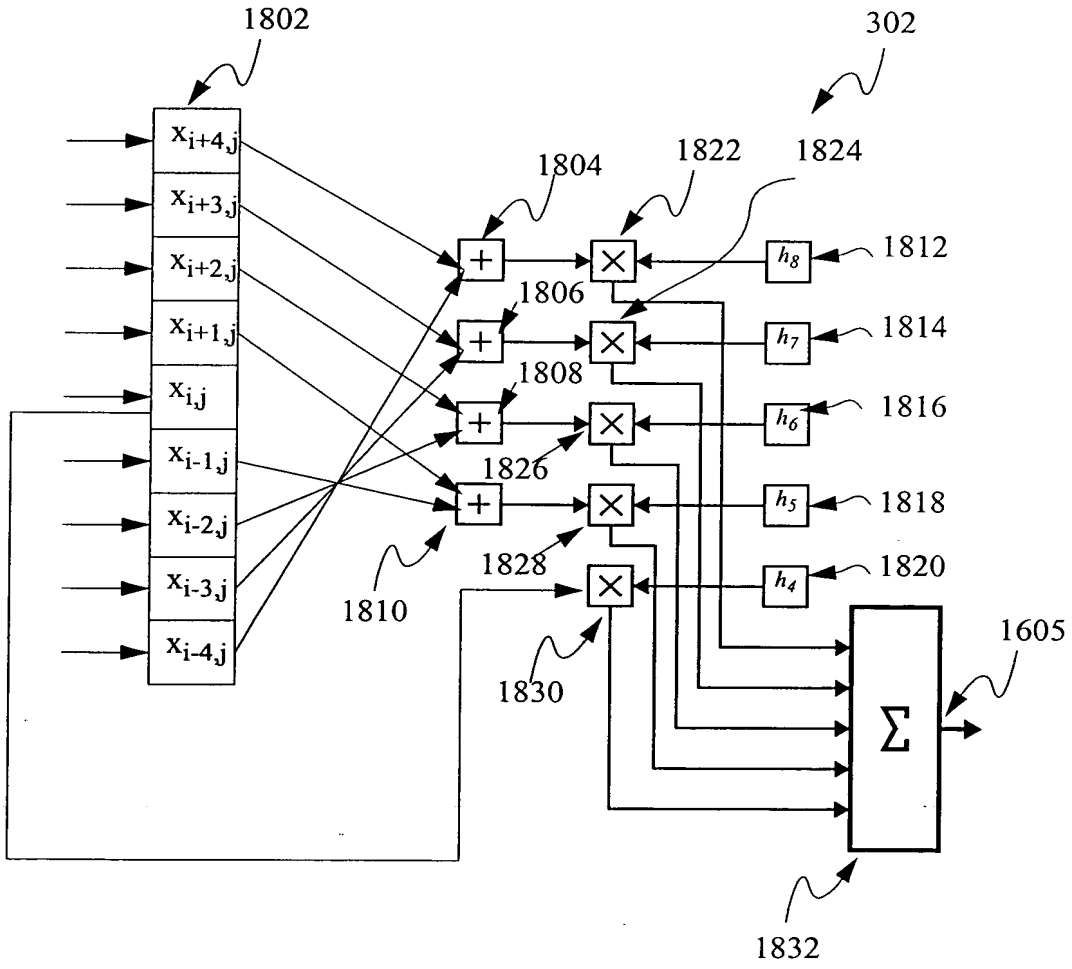

1802
302
1804
1822
1824
1806
1808
1826
1828
1830
1810
1812
1814
1816
1818
1820
1605
1832
$x_{i+4,j}$
$x_{i+3,j}$
$x_{i+2,j}$
$x_{i+1,j}$
$x_{i,j}$
$x_{i-1,j}$
$x_{i-2,j}$
$x_{i-3,j}$
$x_{i-4,j}$
h_8
h_7
h_6
h_5
h_4
Σ

Fig. 15

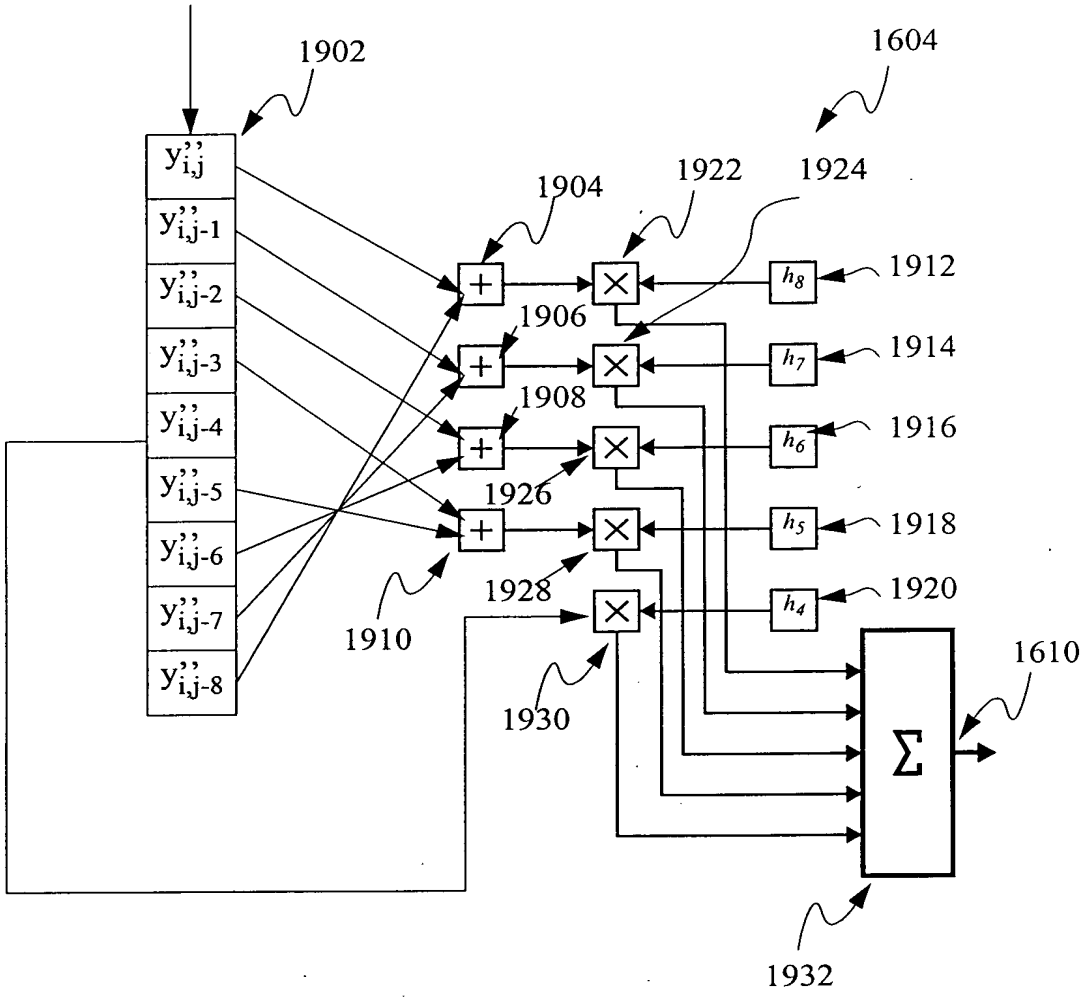

1902
1604
1904
1922
1924
1912
1906
1914
1908
1916
1926
1918
1928
1920
1910
1610
1930
1932
$y''_{i,j}$
$y''_{i,j-1}$
$y''_{i,j-2}$
$y''_{i,j-3}$
$y''_{i,j-4}$
$y''_{i,j-5}$
$y''_{i,j-6}$
$y''_{i,j-7}$
$y''_{i,j-8}$
h_8
h_7
h_6
h_5
h_4
Σ

Fig. 16

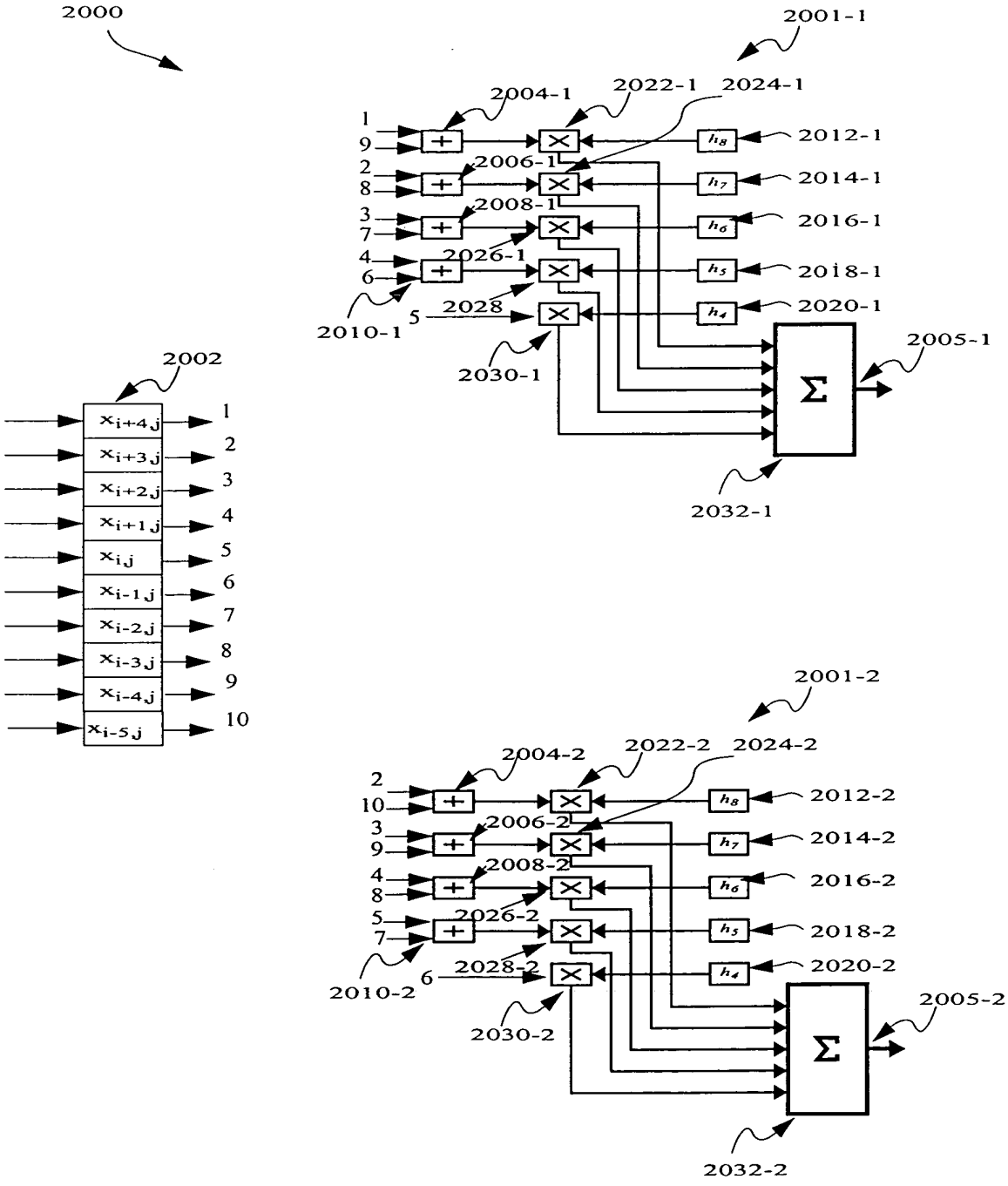

2000
2001-1
2004-1
2022-1
2024-1
2012-1
2006-1
2014-1
2008-1
2016-1
2026-1
2018-1
2028
2020-1
2010-1
2030-1
2005-1
2032-1
2002
2001-2
2004-2
2022-2
2024-2
2012-2
2006-2
2014-2
2008-2
2016-2
2026-2
2018-2
2028-2
2020-2
2010-2
2030-2
2005-2
2032-2

Fig. 17

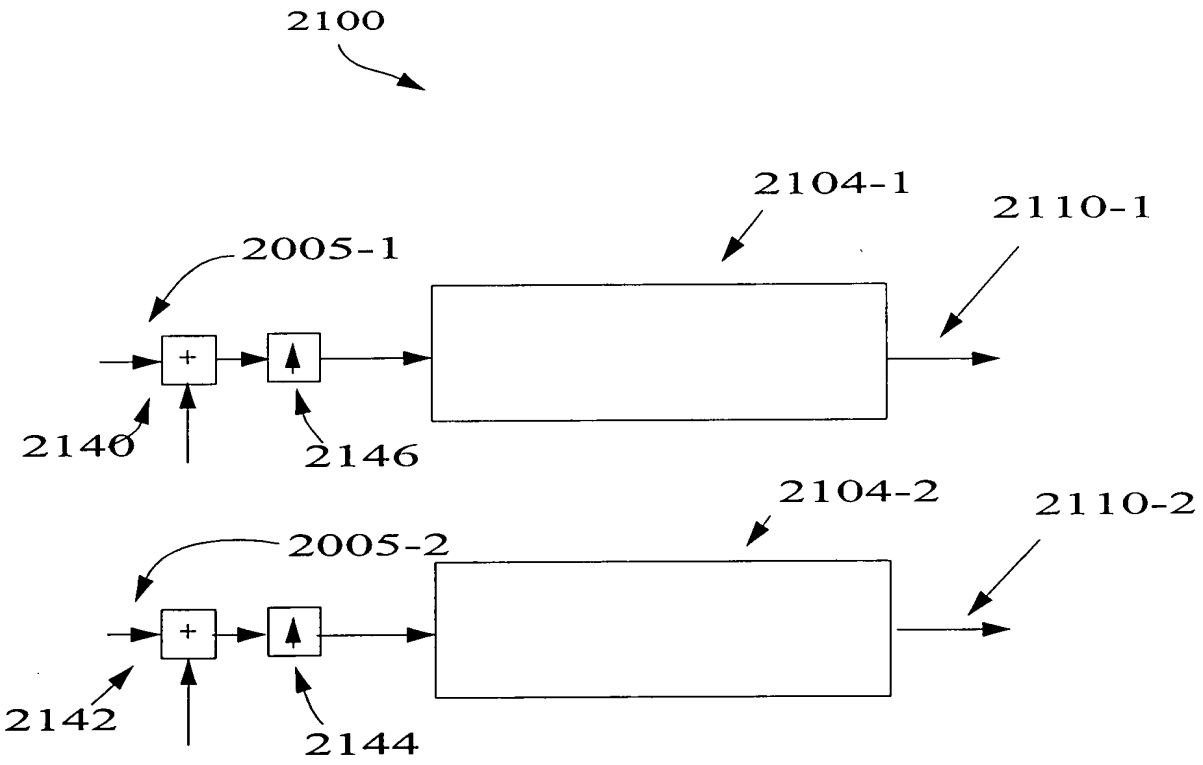

2100
2104-1
2110-1
2005-1
2140
2146
2104-2
2110-2
2005-2
2142
2144

Fig. 18

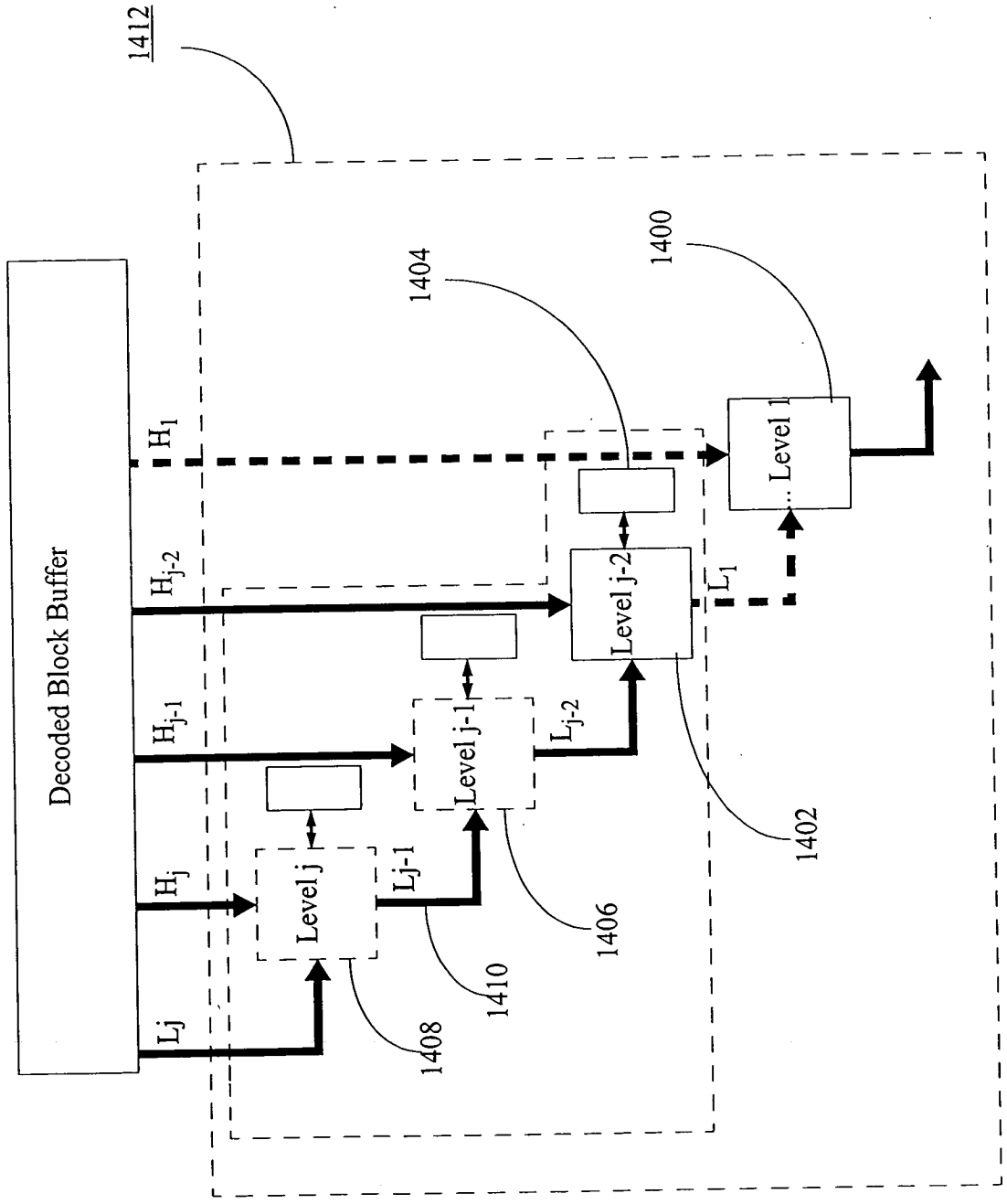

1412
1400
1404
Level 1
Level j-2
Level j-1
Level j
Decoded Block Buffer
H_1
H_{j-2}
H_{j-1}
H_j
L_j
L_{j-1}
L_{j-2}
L_1
1402
1406
1410
1408

Fig. 19

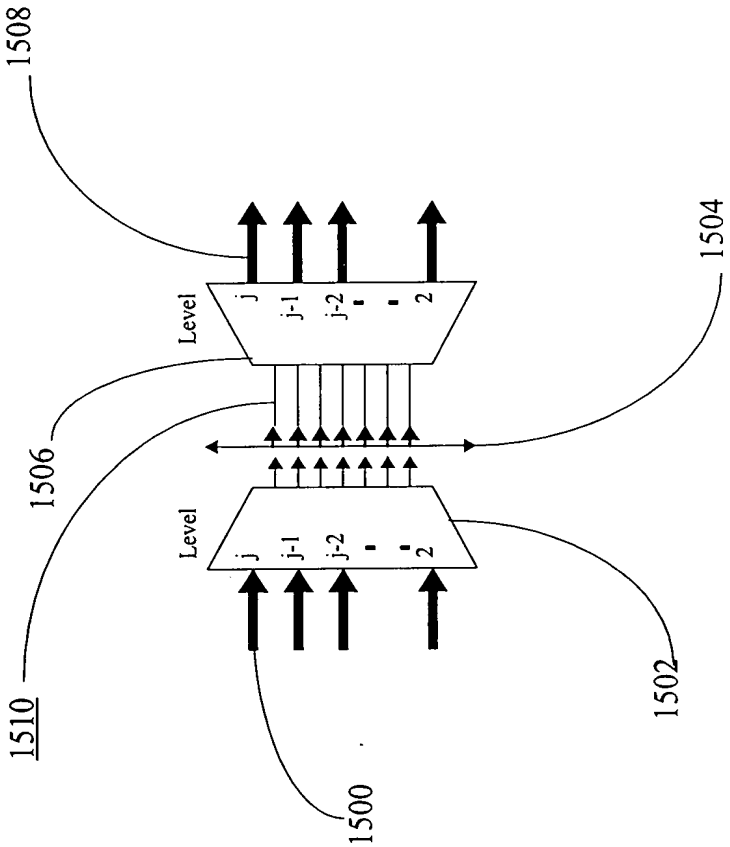


Fig. 20

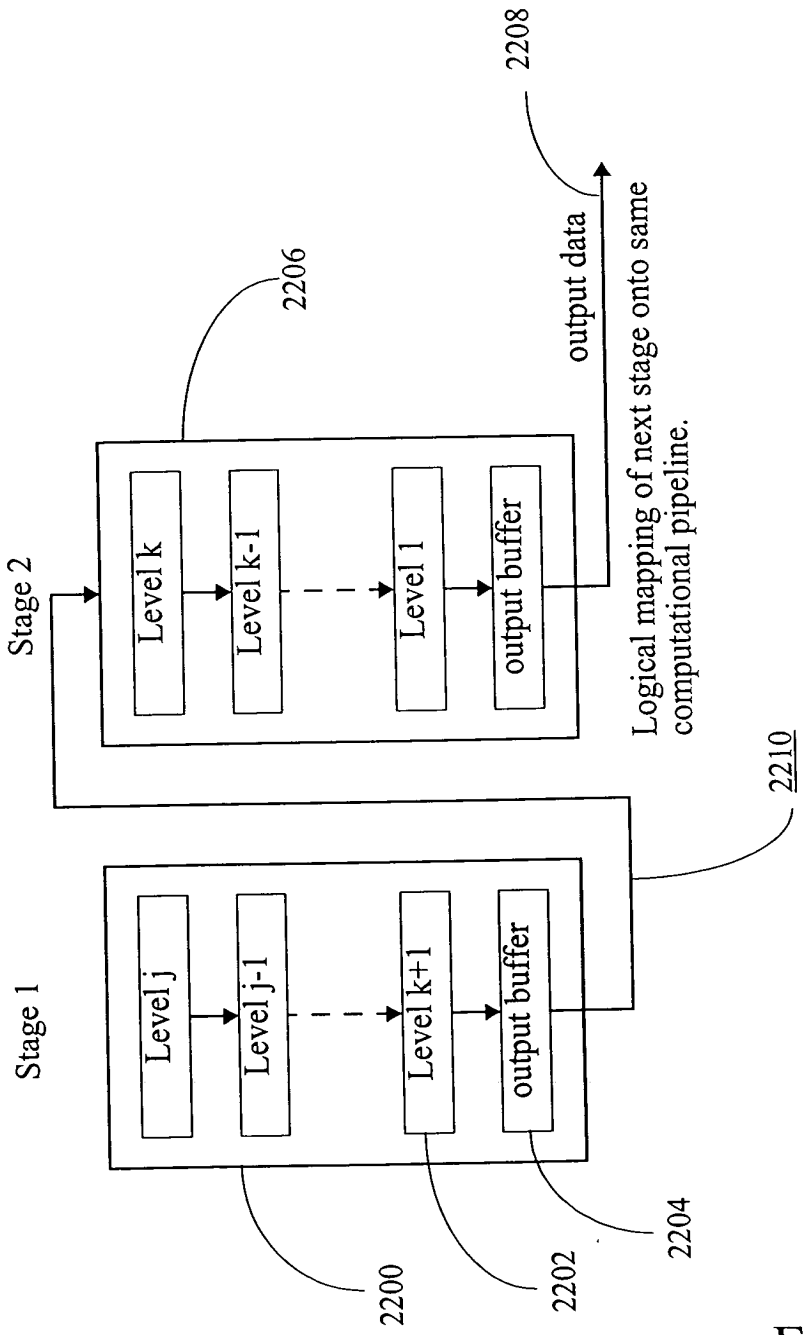

2208
output data
2206
Logical mapping of next stage onto same computational pipeline.
Stage 2
Level k
Level k-1
Level 1
output buffer
2210
Stage 1
Level j
Level j-1
Level k+1
output buffer
2200
2202
2204

Fig. 21

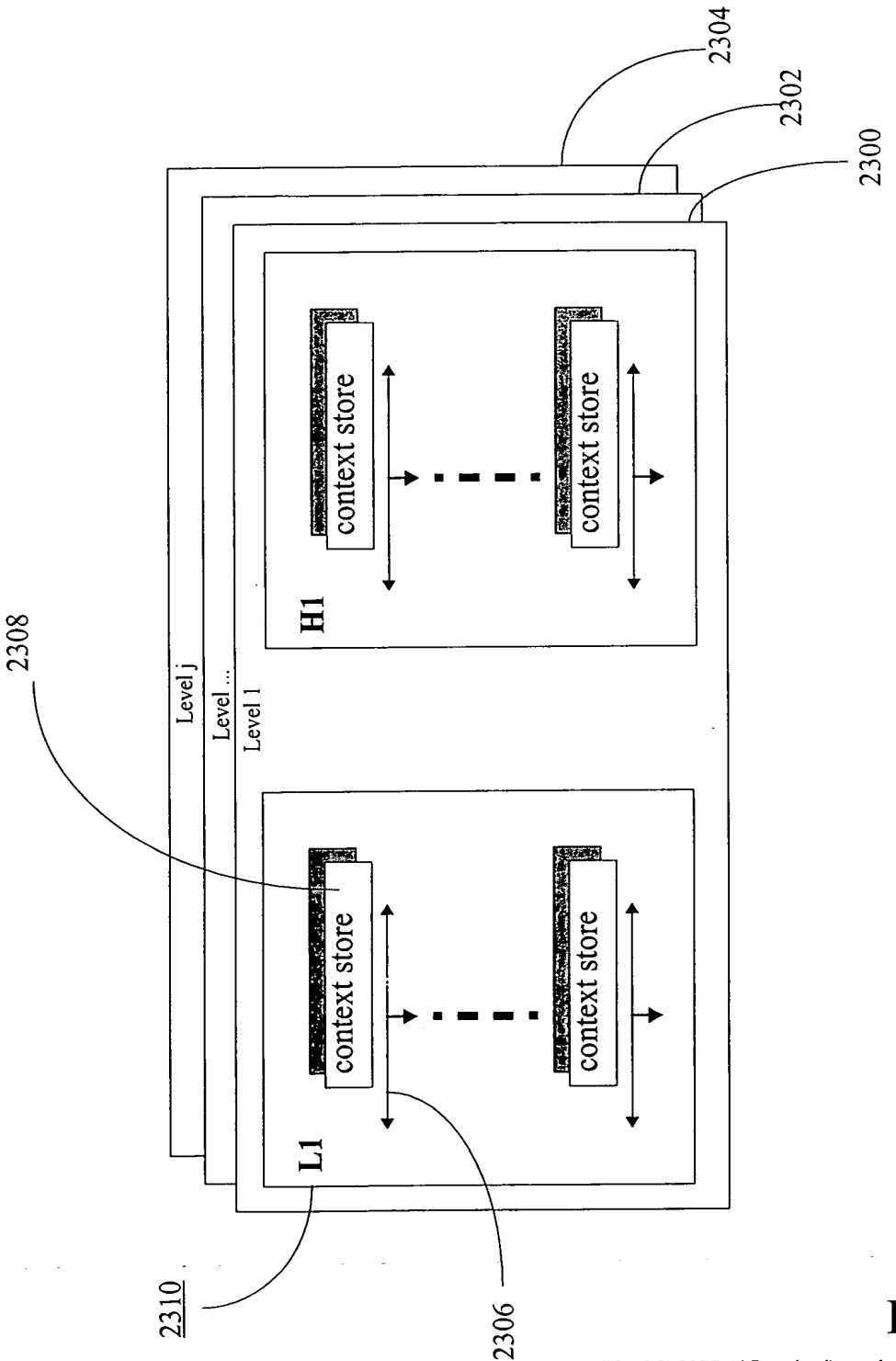

2304
2302
2300
2308
Level j
Level ...
Level 1
H1
context store
context store
L1
context store
context store
2310
2306

Fig. 22

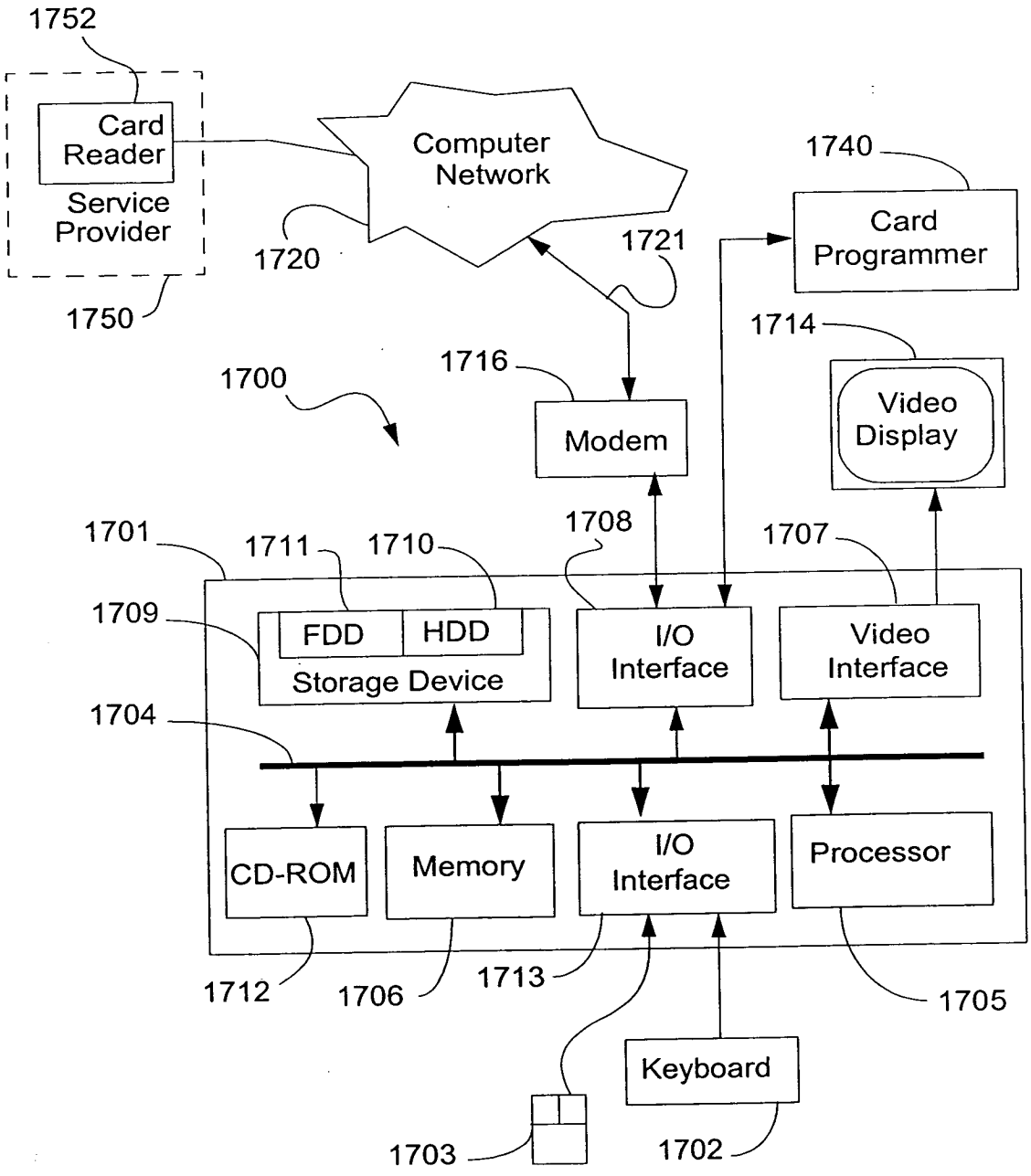

1752
Card Reader
Service Provider
1750
Computer Network
1720
1721
1740
Card Programmer
1714
Video Display
1716
1700
Modem
1708
1701
1711
1710
1707
1709
FDD
HDD
Storage Device
I/O Interface
Video Interface
1704
CD-ROM
Memory
I/O Interface
Processor
1712
1706
1713
1705
Keyboard
1703
1702

Fig. 23